ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DEVELOPING A CONTENT-BASED IMAGE RETRIEVAL SYSTEM USING SEGMENTATION AND COLOR FEATURE

A thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment for the Degree of Master of Science in Computer Engineering

By

Shewatatek Lemma Tena

Advisor

Dr. Kumudha Raimond

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEVELOPING A CONTENT-BASED IMAGE RETRIEVAL SYSTEM USING SEGMENTATION AND COLOR FEATURE

By

Shewatatek Lemma Tena

FACULTY OF TECHNOLOGY

APPROVAL BY BOARD OF EXAMINERS

Chairman, Dept. of Graduate Committee

_____ _____

Signature

Dr. Kumudha Raimond _____

Advisor Signature

Dr. Getachew Alemu _____

Internal Examiner Signature

Dr. Manoj _____

External Examiner Signature

Declaration

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

Name: _Shewatatek Lemma Tena Signature: _____

Place: Addis Ababa

Date of submission: March 2008

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Kumudha Raimond Signature: _____
Advisor's Name

Acknowledgement

First of all, I would like to thank the Almighty God for what He has done for me.
I am highly indebted to my advisor Dr. Kumudha Raimond for her continuous advice and courage. Without her constant encouragement and advice this work will not be completed.

I would like to thank all that helped me in my work. Especially I would like to thank my brother Frezewd Lemma for his advice.

Abstract

In this thesis, a Content-Based Image Retrieval (CBIR) system that is based on segmentation and color feature is presented. CBIR helps to organize and retrieve digital images using their visual content from large databases. The CBIR system proposed in this work consists of segmentation, feature extraction and similarity measuring techniques to store and to retrieve the images.

The segmentation is done using the K-Means clustering algorithm. Unlike existing systems, the segmentation used is unreliable that the image is divided into many small regions that are far from representing semantic objects.

Feature extraction is done for each region of the image. A region is represented by its average color and number of pixels. The average color is calculated for each channel of the RGB color space. An image is stored in the database as a collection of regions.

The similarity of two images is based on the similarity of their regions. Two different similarity measures are proposed in this work. One is based on count of similar regions and the other measure is based on the sum of pixels of the similar regions. Region similarity is based on the Euclidean distance between their average colors and the number of pixels of the regions. Two images that have more number of similar regions are said to be more similar.

The proposed system is compared with some existing systems like **Earth Mover's Distance** (EMD) and SIMPLIcity using three parameters: precision, average rank and standard deviation of the ranks. The performance of the proposed system is promising and found to be better than EMD. However, the system didn't perform well compared to SIMPLIcity, owing to the fact that SIMPLIcity uses color, shape and texture for the feature extraction technique whereas the proposed system uses only the color feature.

Contents

Acronym

BIC	**Border/Interior Pixel Classification**
CBIR	**Content-Based Image Retrieval**
CCV	**Color Coherence Vector**
CIE	**Commission Internationale de L'Eclairage**
DBMS	**Database Management System**
EM	**Expectation Maximization**
EMD	**Earth Mover's Distance**
GUI	**Graphical User Interface**
HSB	**Hue, Saturation and Brightness**
HSV	**Hue, Saturation and Value**
JD	**Jeffrey-Divergence**
KL	**Kullback-Leibler**
MDS	**multi-dimensional scaling**
QF	**Quadratic Form**
RGB	**Red, Green, Blue**
SOM	**Self-Organizing Map**
UC	**University of Colombia**
WWW	**World Wide Web**

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Introduction

The use of images for human communication starts in our cave-dwelling ancestors painting pictures on the walls of their caves. The use of maps and building plans to convey information dates back to pre-Roman times [1].

With the advancement of technology, capturing and communicating image data becomes easier. The invention of photography and television are examples of such technologies. With the invention of computers and digital cameras, capturing, processing and storing of digital images become easier and the number of images increases at an alarming rate.

A digital image is an array, or a matrix, of square pixels (picture elements) arranged in columns and rows. [2]

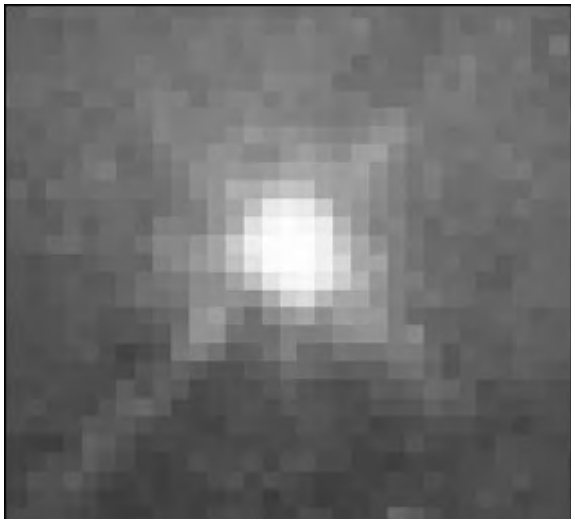


Figure 1.1 An image — an array or a matrix of pixels arranged in columns and rows.

In a (8-bit) greyscale image each picture element has an assigned intensity that ranges from 0 to 255. A grey scale image is what people normally call a black and white image, but the name emphasizes that such an image will also include many shades of grey.

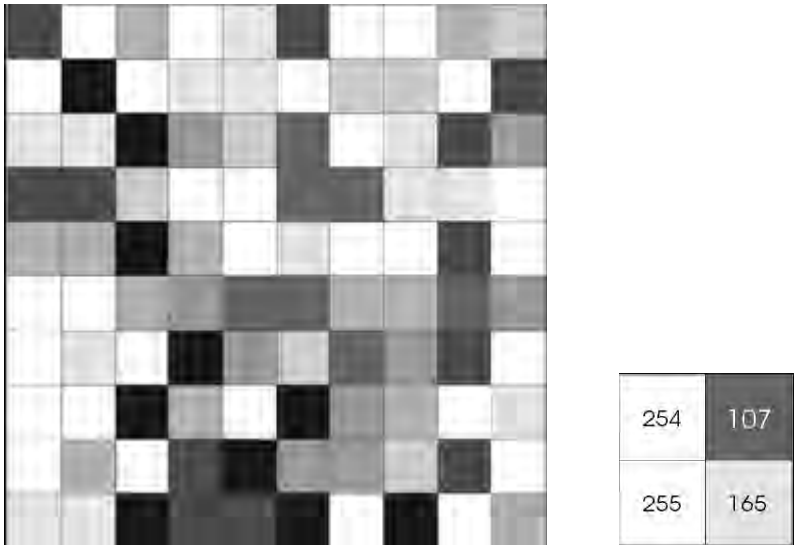


Figure 1.2 Each pixel has a value from 0 (black) to 255 (white). The possible range of the pixel values depend on the color depth of the image, here 8 bit = 256 tones or grayscales.

A normal grayscale image has 8 bit color depth = 256 grayscales. A “true color” image has 24 bit color depth = 8 x 8 x 8 bits = 256 x 256 x 256 colors = ~16 million colors. [2]

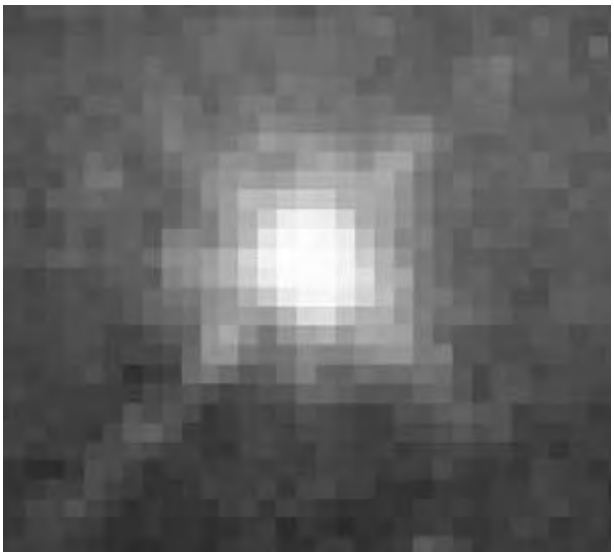


Figure 1.3 A true-color image assembled from three grayscale images colored red, green and blue. Such an image may contain up to 16 million different colors.

Some of the most common file formats are: [2]

GIF — an 8-bit (256 color), non-destructively compressed bitmap format. Mostly used for web. Has several sub-standards one of which is the animated GIF.

JPEG — a very efficient (i.e. much information per byte) destructively compressed 24 bit (16 million colors) bitmap format. Widely used, especially for web and Internet (bandwidth-limited).

TIFF — the standard 24 bit publication bitmap format. Compresses non-destructively with, for instance, Lempel-Ziv-Welch (LZW) compression.

PS — Postscript, a standard vector format. Has numerous sub-standards and can be difficult to transport across platforms and operating systems.

PSD – a dedicated Photoshop format that keeps all the information in an image including all the layers.

The creation of the World Wide Web (WWW), enabling users to access data in a variant of media format, is used as a stimulus for organizations having large image collection to convert their collection to digital format. Currently, the number of digital images on the WWW is estimated in 100 of millions.

As the number of digital images increases, it becomes clear that there should be a way for efficient storage and retrieval of images. Earlier approaches for the problem were based on text annotation of the image. To every image in the collection, a text description will be added manually. The search is done based on this text description. Even today image searches on the WWW are mostly based on text searches. (E.g. Google and Yahoo image searches).

Text-based search of images has a number of drawbacks: [3]

1. Some images are too complex to describe in a short text. (A picture speaks more than thousand words)

2. It requires humans to write the description.

3. There is the prospect of having to re-index sections of the collection; for example if a news event makes a previously unknown person famous.

As the text-based system is not a suitable system for image search, we have to find another way that removes the drawbacks of the text-based search.

1.2. Background of the problem

The other option to search in large image databases is to use their visual content. Clearly, a system that searches images by their visual content will not have the drawbacks of text based systems listed above. For example, there is no need of manual work as in the case of text description search, because the search is based on the visual content of the image itself. The searching of images by their visual content like color, texture and shape is called CBIR [1]. CBIR is a technology that helps us to organize digital pictures and retrieve them with their visual content.

To search images by their content, two things have to be done. [4]

1. The image has to be re-encoded into some mathematical form and stored in a database.

2. There should be a mechanism to compare these mathematical forms.

Re-encoding is needed because an image is a collection of pixels with no meaning by itself. There is a gap between the visual information conveyed by the image and the way it is encoded. The process of re-encoding the image into a mathematical form suitable for comparison purpose is called ***feature extraction***. The mathematical form of the image is called its ***signature***. The components of a signature are called ***features***. For example, a color moment is a signature constructed from color feature. A feature captures a certain visual property of an image, either globally for the entire image, or locally for a small group of pixels. Features extracted for the whole image are called ***global features.*** Features extracted for divided subparts of the image are called ***local features***.

Features can also be grouped as ***low-level*** and ***high-level features.***

Low-level features are features that can be obtained from the pixel itself. Examples are color and texture. High-level features are features obtained from the combination of low-

level features. Examples are edge and shape. Some of the most widely used features are discussed below [5].

Color

Examining images based on the colors they contain is one of the most widely used techniques because it does not depend on image size or orientation. Color searches will usually involve comparing color histograms, though this is not the only technique in practice.

Texture

Texture measures look for visual patterns in images and how they are spatially defined. Textures are represented by texels which are then placed into a number of sets, depending on how many textures are detected in the image. These sets not only define the texture, but also where in the image the texture is located.

Shape

Shape does not refer to the shape of an image but to the shape of a particular region that is being sought out. Shapes will often be determined first applying segmentation or edge detection to an image. In some cases accurate shape detection will require human intervention because methods like segmentation are very difficult to completely automate.

Once the signatures of the images are constructed and stored in a database, the next step will be to compare these signatures. ***Similarity measures*** are methods that are used to compare signatures of images. They are used to compare the signature of the query image with that of signatures of images in the database. Images whose signatures are more similar to the query image will be retrieved.

In short, CBIR systems have the following two components [6]:

1. **Feature extraction:** In this step the image is re-encoded using the low level features and stored in a feature database.

2. **Similarity measurement:** In CBIR systems, the user presents a sample image and the system returns images whose content is similar to the sample. Here there is a measurement of similarity between the sample image and images in the database.

Figure 1.4 shows the CBIR process.

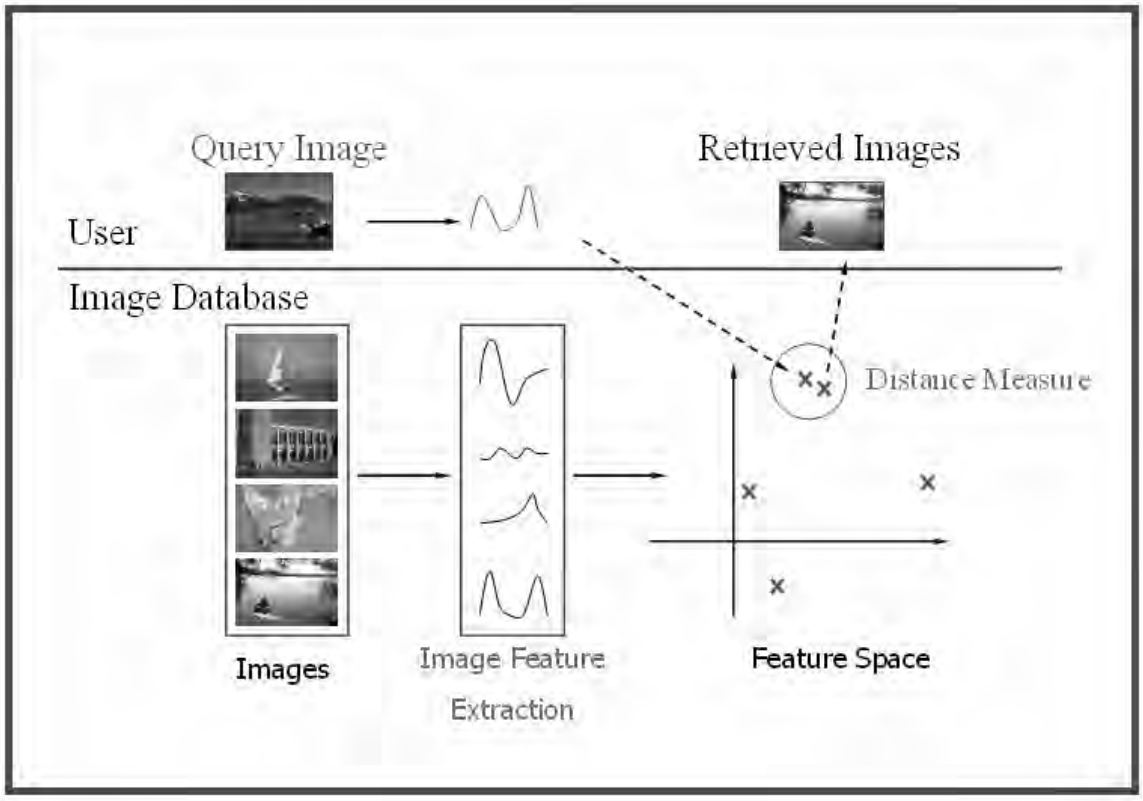


Figure 1.4 CBIR Process

In short, CBIR addresses the problem of finding images relevant to the users' information needs from image databases, based principally on low-level visual features for which automatic extraction methods are available. Due to the inherently weak connection between the high-level semantic concepts that humans naturally associate with images and the low-level visual features that the computer is relying upon, the task of developing this kind of system is very challenging.

1.3. Motivation

A number of studies have been done on the area of content-based information retrieval in general and CBIR in particular. Based on the findings of these studies, both commercial and academic CBIR systems have been developed. A good survey is found in [7].

The main problem in CBIR is how to define the content of the image based on low-level features like color, texture and shape and to define the similarity measure. It is very challenging to extract visual information based on these low level features. Researches have continued to come up with a system that works for images of any type.

Thus far, a large amount of research was done within the area of CBIR that deals with full-image retrieval. Full image retrieval systems use the global feature of images. In these systems, both the query image and the images in the database are not segmented into regions, and the features are extracted for the whole image. An example is [8].

A different type of CBIR is one tried by [9]. Here the system searches for images that contain the query image, i.e., images where the query image is part of the overall image will be retrieved. In this system, the images in the database are segmented but the query image is not segmented. So, for the query image, the global features are used whereas for images in the database, the local features are used.

Another system is what is called region-based image retrieval [10]. In this approach, the user specifies an image part of his interest and retrieves visually similar parts in other images. The part of the image that is not the interest of the user is called the background and is not considered in the retrieval process. Here, local features are used for both the query image and the images in the database. Although both the query image and the images in the database are segmented, only one part of the query image is used for searching. The remaining parts are taken as background and not used for the searching.

The other type of region-based CBIR system uses local features for both the query image and the images in the database. All the regions in the query image are used for the searching. Examples for such systems are [11] and [12].

Most of the existing CBIR systems are region-based systems because region-based systems are better than full-image retrieval systems. Region-based systems use different segmentation techniques to divide images into subparts. The current CBIR systems use segmentation techniques that try to divide the image into semantic objects [4].

It is said by RITENDRA DATTA, DHIRAJ JOSHI, JIA LI, and JAMES Z. WANG in their survey of CBIR in the ACM Transactions on Computing Surveys (2007), reference [4],

> "While there is no denying that achieving good segmentation is a major step toward image understanding, some issues plaguing current techniques are computational complexity, reliability of good segmentation, and acceptable segmentation quality assessment methods."

Some of the possible solutions suggested by the above paper are

- to reduce dependence on reliable segmentation
- to involve every generated segment of an image in the matching process

Based on these suggestions, a system is developed that

- uses unreliable segmentation technique to divide the image into regions
- incorporates almost all regions in the matching process

1.4. Objectives

The main objective of this thesis is to develop a CBIR system based on color feature of segmented parts of the image.

Specific objectives are the following

1. Studying about the different image segmentation techniques and selecting the one that best suits based on its efficiency and effectiveness

2. Studying about the different color feature extraction methods and selecting the one that is appropriate.

3. Developing a new similarity measure based on color feature and segmented images

4. Developing a complete program that implements the features selected in the above steps.

1.5 Scope of Study

In this study, it is tried to review the different techniques of segmentation and the different algorithms for color feature extraction. In addition, a new similarity algorithm is proposed. A complete computer program that implements the selected segmentation technique and color feature extraction algorithm and the new similarity algorithm is developed.

The block diagram representation of the CBIR system is given below in Fig 1.2. As shown in Figure 1.5, the system has two processes. In the ***image addition*** process, an image is added to the database. First it is segmented into regions and then for each region a feature is extracted. From the features of the regions, a signature is constructed and added to the signature database. In the ***retrieval process,*** a query image is supplied by the user and the signature is constructed for the query image by a similar process in which, the signatures for the images in the database is constructed. Then a similarity measure is used to compare signatures of images in the database and that of the query image. Finally images whose signature is more similar to the query image will be returned as an output.

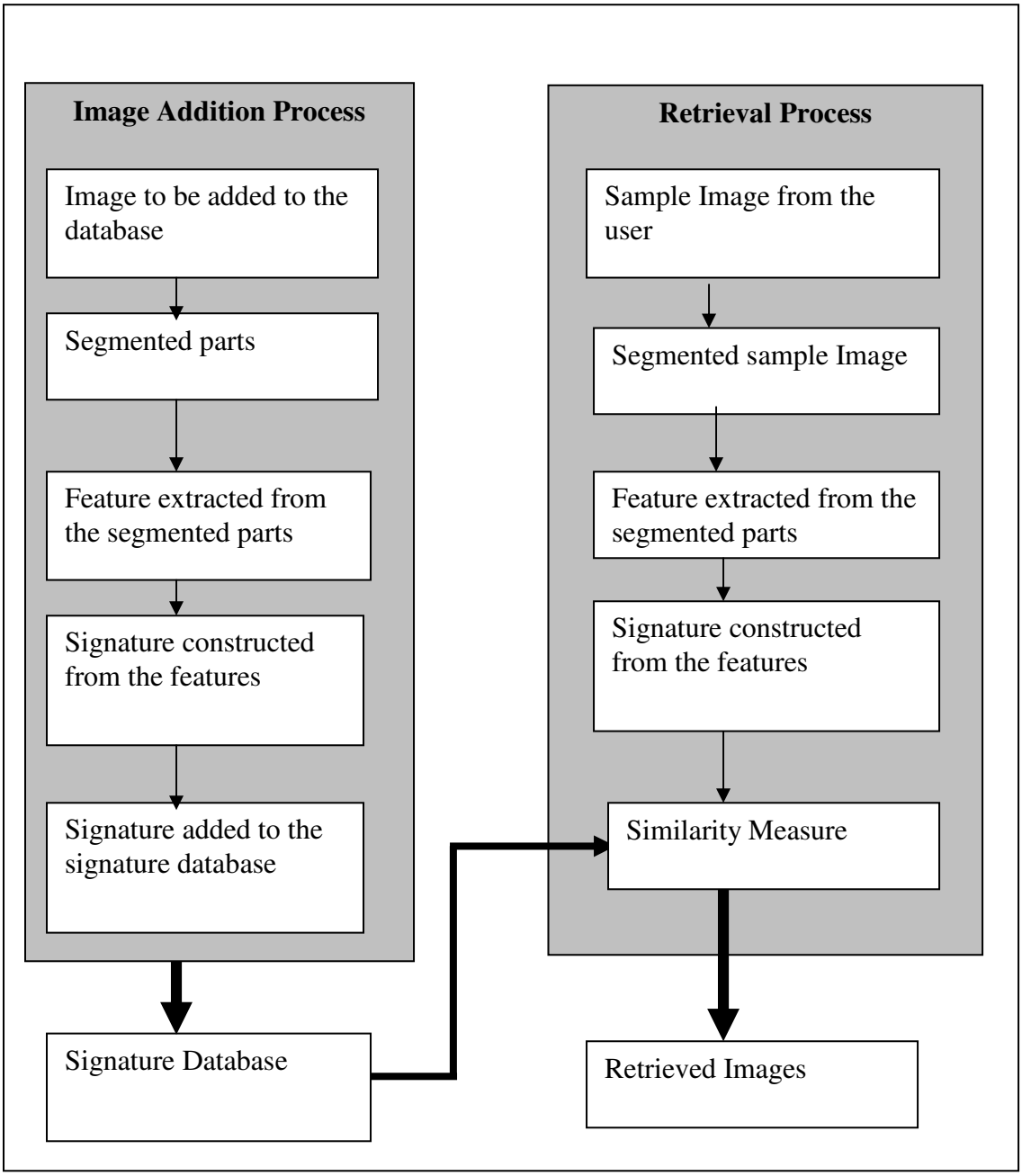


Figure 1.5 Block Diagram of the system to be developed

1.6 Thesis Outline

The organization of the paper is as follows: Chapter 2 discusses the existing techniques on image segmentation, feature extraction and similarity measures. Chapter 3 discusses the analysis and the implementation details of segmentation. Chapter 4 discusses the analysis and the implementation details of the feature extraction step. Chapter 5 discusses about the results obtained from the new system. Performance measurement is the essence of chapter 6. Conclusion and further work is discussed in Chapter 7.

Chapter 2

Literature Survey

As shown in the block diagram in Figure 1.2, the system developed has the following main parts

- Image segmentation
- Extraction of features
- Similarity measures

In this chapter, the different techniques of image segmentation, feature extraction and similarity measures will be discussed. In addition, some of the existing CBIR systems and the image database used for testing the performance of the proposed system will be explored.

The chapter is organized as follows: Section 2.1 discusses the different color image segmentation techniques. Section 2.2 discusses the different color features and the similarity measures are discussed in section 2.4. Section 2.5 discusses about the image database used for testing the performance of the system. A conclusion is drawn in section 2.6.

2.1 Color Image Segmentation – State of the art

Image segmentation is the process of dividing an image into regions such that each region is homogeneous with respect to some characteristic. A formal definition of image segmentation is as follows: [13]

> If P( ) is a homogeneity predicate defined on groups of connected pixels, then segmentation is a partition of the set F into connected subsets or regions (S_1, S_2, …,S_n) such that

$$\bigcup_{i=1}^{n} S_i = F, \quad \text{with } S_i \cap S_j = \Phi, \qquad (i \neq j) \tag{2.1}$$

The uniformity predicate P (S_i) = true for all regions, S_i, and P (S_i U S_j) = false, when $i \neq j$ and S_i and S_j are neighbors.

The homogeneity predicate can be any feature of an image like color and texture.

According to [14], "the image segmentation problem is basically one of psychophysical perception and therefore not susceptible to a purely analytical solution". According to [15], there is no theory of image segmentation. As a consequence, no single standard method of image segmentation has emerged. Rather, there are a collection of ad hoc methods that have received some degree of popularity.

There are a number of surveys that try to categorize the different segmentation techniques. According to [16], segmentation techniques can be grouped into the following categories.

- Histogram Thresholding and Color Space Clustering
- Region Based Approaches
- Edge Detection
- Fuzzy Techniques
- Physics Based Approaches and
- Neural Networks Approaches

Segmentation algorithms are categorized in [17] as early, middle and recent and the recent segmentation techniques are grouped as follows.

- Feature–based techniques.
    - Clustering.
    - Adaptive k–mean clustering.
    - Histogram thresholding.
- Image–based techniques.
    - Split & Merge.
    - Region–growing.
    - Graph–theoretical techniques.
    - Edge–based techniques.
    - Neural networks.

- Physics–based techniques.

As can be seen, the two surveys are more or less similar except that graph–theoretical techniques are not included in the first survey. In addition, clustering is further divided into two groups in the latter survey. The descriptions, advantages and disadvantages of the different algorithms are given in the above surveys as listed in Table 2.1 below.

Segmentation Technique	**Method Description**	**Advantages**	**Disadvantages**
Histogram Thresholding	Requires that the histogram of an image has a number of peaks, each corresponds to a region	It does not need prior information of the image. For a wide class of images satisfying the requirement, this method works very well with low computation complexity.	1. It does not work well for an image without any obvious peaks or with broad and flat valleys. 2. Does not consider the spatial details, so cannot guarantee that the segmented regions are contiguous.
Feature Space clustering	Assumes that each region in the image forms a separate cluster in the feature space. Can be generally broken into two steps: 1) categorize the points in the feature space into clusters; 2) map the clusters back to the spatial domain to form separate regions.	Straightforward for classification and easy for implementation	1. how to determine the number of clusters (known as cluster validity) 2. features are often image dependent and how to select features so as to obtain satisfactory segmentation results remain unclear 3. does not utilize spatial information

Segmentation Technique	**Method Description**	**Advantages**	**Disadvantages**
Region-based Approaches	Group pixels into homogeneous regions. Including region growing, region splitting, region merging or their combination	Work best when the region homogeneity criterion is easy to define. They are also more noise-immune than edge detection approach	1. Are by nature sequential and quite expensive both in computational time and memory 2. Region growing has inherent dependency on the selection of seed region and the order in which pixels and regions are examined 3. The resulting segments by region splitting appear too square due to the splitting scheme
Edge detection approaches	Based on the detection of discontinuity, normally tries to locate points with more or less abrupt changes in grey level. Usually classified into two categories: sequential and parallel	Edge detection technique is the way in which human perceives objects and works well for images having good contrast between regions.	1. Does not work well with images in which the edges are ill-defined or there are too many edges 2. It is not a trivial job to produce a closed curve or boundary 3. Less immune to noise than other techniques, e.g. thresholding and clustering

Segmentation Technique	Method Description	Advantages	Disadvantages
Fuzzy Approaches	Apply fuzzy operators, properties, mathematics, and inference rules(IF-THEN rules), provide a way to handle the uncertainty inherent in a variety of problems due to ambiguity rather than randomness	Fuzzy membership function can be used to represent the degree of some properties or linguistic phrase, and fuzzy IF-THEN rules can be used to perform approximate inference	1. determination of fuzzy membership is not a trivial job 2. the computation involved in fuzzy approaches could be intensive
Neural network approaches	Using neural networks to perform classification or clustering	No need to write complicated programs. Can fully utilize the parallel nature of neural networks	1. Training time is long 2. Initialization may affect the results 3. overtraining should be avoided
Physics Based Techniques	Employs physical models to locate the objects boundaries while eliminating the spurious edges of shadow or highlights in a color image	Alleged robustness to highlights, shadowing, and shades Segment surfaces by their material composition	• Restriction to one or two types of material, such as dielectrics or metals • Difficulty to identify highlights and shades in real images • Some algorithms need shape information, not always available • Computationally expensive algorithms
Graph–Theoretical Techniques	Partition a graph describing the whole image into a set of connected components that correspond to image regions.	• Very good translation of spatial and feature relations into graph–based representations • Some greedy approaches provide very fast algorithms	• Some approaches are extremely time consuming • Local features may be imposed over global ones

Table 2.1 Segmentation Techniques

As shown in Table 2.1, all the segmentation techniques have their own advantages and disadvantages. It is impossible to get a segmentation technique that works equally well for all the images types. In this thesis work, the segmentation technique need not be of high performance, because segmentation is needed to divide the image into homogeneous regions that are far from representing semantic objects. Hence speed is given priority. Therefore the fastest segmentation algorithm is selected. K-Means clustering algorithm is popular for its speed [4]. Hence K-Means segmentation is used to segment images.

Any feature of an image can be used as a homogeneity predicate during the segmentation process. Here the feature used for segmenting the image is the color feature. Although features like texture can be used for the segmentation, they are not used here leaving them for further study.

2.2 Color Feature Extraction

The next step after segmentation is feature extraction. As it is already said a feature is defined to capture some visual property of an image. A signature of an image is obtained from features through a process called feature extraction. Another name for signature is feature vector. [18] Most CBIR systems use color, texture and shape features. In this work, only the color feature will be used leaving the other features for further study.

According to [18], research on color based systems can be grouped into three sub-areas:

- Definition of adequate color space
- Appropriate extraction algorithm
- Similarity measures

The different similarity measures will be discussed in section 2.3. In this section, color space and extraction algorithms are discussed.

2.2.1 Color Spaces

Color is a property that depends on the reflection of light to the eye and the processing of that information in the brain. We use color everyday to tell the difference between objects, places, and the time of day.

The characteristics used to distinguish one color from another are: [19]

Hue: is an attribute associate with the dominant wavelength in a mixture of light waves. It represents the dominant color as perceived by observer (for example, orange, red, pink, etc)

Saturation: refers to the relative purity or the amount of white light mixed with a hue. Pure colors are fully saturated. Colors such as pink (red and white) are less saturated with the saturation being inversely proportioned to the amount of white light added.

Brightness: is a measure of the intensity of light. It is this property of color that helps us to differentiate between dim and bright.

The properties *hue* and *saturation* are said to jointly describe the *chromaticity* of the color.

Usually colors are defined in three dimensional color spaces [20] [21], [22]. The most commonly used color spaces are listed below [20].

RGB

The RGB color space describes a color in terms of the magnitude of three light ingredients (channels) of different specified chromaticity which if added together will produce that color. These three "primaries" are described as red, green, and blue and the variables R, G, and B describe the amounts of each in the "mix".

HSB

HSB stands for hue, saturation and brightness. It is also called HSV for hue, saturation and value. The definition of these components is as given above.

CIE XYZ

CIE (Commission Internationale de L'Eclairage) XYZ is a complete model developed in 1931 that can recreate all colors in the visible spectrum. XYZ defines virtual primary colors X, Y, and Z, which are the wavelengths that the rods and cones in the human eye are most sensitive to. The Y primary was specifically designed to follow the luminous efficiency function of human eyes.

CIE L*a*b*

In 1976, the CIE created a refined model of XYZ called L*a*b*. In L*a*b*, L is the luminance, a* is a value for which -a* is green, and +a* is red, b* is a value for which -b* is blue, and +b* is yellow.

Use the following formula to convert from CIE XYZ to CIE L*a*b*:

L* = -16 + 116 * (pow(Y / Yn), 1/3)

a* = 500 * (pow(X / Xn, 1/3) - pow(Y / Yn, 1/3)

b* = 200 * (pow(Y / Yn, 1/3) - pow(Z / Zn, 1/3) 2.2

Where X, Y, Z exist on the scale of 0, 0, 0 to Xn, Yn, Zn.

There has been no consensus about which color feature space is most suitable for color image retrieval. The problem is that there is no universally accepted color space, and color perception is significantly subjective [22]. In this work, RGB is used as the color space.

2.2.2 Extraction algorithms

In general color features can be grouped into two: those that do not include spatial distribution and those that include spatial distribution. [18] Examples of the first category are color histogram [23] and color moments [24]. Examples of the second category are Color Coherence Vector (CCV) [25], Border/Interior Pixel Classification (BIC) [26], and Color Correlogram [27].

Features that include spatial distribution are meaningful when the feature is extracted for the whole image, i.e. when global feature is used. In this work, feature is extracted for regions of the image, hence only the features that do not include spatial distribution are considered. Color Histogram and color moments are discussed below.

Color Histogram [24]

A color histogram is formed by discretizing the colors within an image and counting the number of colors that fall into each bin. Since the typical computer represents color images with up to 16,777,216 colors, this process generally involves substantial

quantization of the color space. The main issues regarding the color histograms are the choice of color space, the type of quantization and the degree of quantization. When a perceptually uniform color space is chosen, uniform quantization is appropriate. If a non-uniform color space is chosen, then non-uniform quantization may be needed.

Color histograms are easy to compute, and they are invariant to the rotation and translation of image content. However, color histograms have several inherent problems for the task of image indexing and retrieval. The first concern is their sensitivity to noisy interference such as lighting intensity changes and quantization errors. The second problem is their high dimensionality on representation. Even with coarse quantization over a chosen color space, color histogram feature spaces often occupy more than one hundred dimensions (i.e., histogram bins) which significantly increases the computation of distance measurement on the retrieval stage.

Another variant of color histogram is the one that uses fuzzy techniques. It is discussed in [28].

Color Moments [23]

Color moments are measures that can be used to differentiate images based on their features of color. Once calculated, these moments provide a measurement for color similarity between images. These values of similarity can then be compared to the values of images indexed in a database for tasks like image retrieval. The basis of color moments lays in the assumption that the distribution of color in an image can be interpreted as a probability distribution. Probability distributions are characterized by a number of unique moments (e.g. Normal distributions are differentiated by their mean and variance). It therefore follows that if the color in an image follows a certain probability distribution, the moments of that distribution can then be used as features to identify that image based on color.

Stricker and Orengo [23] use three central moments of an image's color distribution. They are Mean, Standard deviation and Skewness. A color can be defined by 3 or more values. Moments are calculated for each of these channels in an image. An image therefore is characterized by 9 moments, 3 moments for each 3 color channels. We will

define the i^{th} color channel at the j^{th} image pixel as p_{ij} . The three color moments can be defined as:

MOMENT 1 – Mean:

$$E_i = \sum_{N}^{j=1} \frac{1}{N} p_{ij} \qquad 2.3$$

Mean can be understood as the average color value in the image.

MOMENT 2 – Standard Deviation:

$$\sigma_i = \sqrt{\left(\frac{1}{N} \sum_{N}^{j=1} (p_{ij} - E_i)^2\right)} \qquad 2.4$$

The standard deviation is the square root of the variance of the distribution.

MOMENT 3 – Skewness:

$$s_i = \sqrt[3]{\left(\frac{1}{N} \sum_{N}^{j=1} (p_{ij} - E_i)^3\right)} \qquad 2.5$$

Skewness can be understood as a measure of the degree of asymmetry in the distribution. N is number of pixels.

Since only 9 (three moments for each of the three color components) numbers are used to represent the color content of each image, color moments are a very compact representation compared to other color features like histogram [24].

The goal of color feature extraction is to obtain compact, perceptually relevant representation of the color content of an image [5]. To make it compact, low dimension features should be used. To make it perceptually relevant high dimensional features should be used. There is a trade-off between compactness and perceptual relevancy.

In this work, an image is divided into a number of regions using unreliable segmentation and feature is extracted for each region. Dividing an image into a number of regions and

representing each region by a compact feature is equivalent to representing the whole image by high dimensional feature. Therefore, a region is represented by its average color and number of pixels and the whole image is represented by the collection of features of its regions. Although the representation is not compact, it is perceptually relevant because more information is incorporated by using high dimensional feature.

2.3 Similarity Measures

Instead of exact matching, CBIR calculates visual similarities between a query image and images in a database. Accordingly, the retrieval result is not a single image but a list of images ranked by their similarities with the query image. Many similarity measures have been developed for image retrieval based on empirical estimates of the distribution of features in recent years. Different *similarity/distance measures* will affect retrieval performances of an image retrieval system significantly. [29]

Some of the similarity measures are discussed below. [29]

Minkowski-Form Distance

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

$$D(I,J)=(\sum_i |f_i(I)-f_i(J)|^p)^{1/p} \qquad 2.6$$

when p=1, 2, and ∞, $D(I, J)$ is the *L1*, *L2* (also called Euclidean distance), and $L\infty$ distance respectively. Minkowski-form distance is the most widely used metric for image retrieval.

The *Histogram intersection* can be taken as a special case of *L1* distance, which is used by Swain and Ballard [24] to compute the similarity between color images. The intersection of the two histograms of I and J is defined as:

$$S(I,J)=\frac{\sum_{i=1}^{N}\min(f_i(I),f_i(J))}{\sum_{i=1}^{N}f_i(J)} \qquad 2.7$$

Quadratic Form (QF) Distance

The Minkowski distance treats all bins of the feature histogram entirely independently and does not account for the fact that certain pairs of bins correspond to features which are perceptually more similar than other pairs. To solve this problem, *quadratic form distance* is introduced:

$$D(I,J) = \sqrt{(\mathbf{F}_I - \mathbf{F}_J)^T \mathbf{A}(\mathbf{F}_I - \mathbf{F}_J)} \qquad 2.8$$

where *A*= [*aij*] is a similarity matrix, and *aij* denotes the similarity between bin *i* and *j*. **F** *I* and **F** *J* are vectors that list all the entries in *fi*(*I*) and *fi*(*J*).

Mahalanobis Distance

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

$$D(I,J) = \sqrt{(\mathbf{F}_I - \mathbf{F}_J)^T C^{-1} (\mathbf{F}_I - \mathbf{F}_J)} \qquad 2.9$$

where *C* is the covariance matrix of the feature vectors. The Mahalanobis distance can be simplified if feature dimensions are independent. In this case, only a variance of each feature component, *ci*, is needed.

$$D(I,J) = \sum_{i=1}^{N} (\mathbf{F}_I - \mathbf{F}_J)^2 / c_i \qquad 2.10$$

Kullback-Leibler (KL) Divergence and Jeffrey-Divergence (JD)

The *Kullback-Leibler (KL) divergence* measures how compact one feature distribution can be coded using the other one as the codebook. The KL divergence between two images *I* and *J* is defined as:

$$D(I,J) = \sum_i f_i(I) \log \frac{f_i(I)}{f_i(J)} \qquad 2.11$$

The *Jeffrey-divergence* (*JD*) is defined by:

$$D(I,J) = \sum_i f_i(I) \log \frac{f_i(I)}{\hat{f}_i} + f_i(J) \log \frac{f_i(J)}{\hat{f}_i} \qquad 2.12$$

where $\hat{f}i = [\, fi\ (I\ ) + fi\ (J\ )]\ /2$. In contrast to KL-divergence, JD is symmetric and numerically more stable when comparing two empirical distributions.

Any of the above similarity measures can be used for measuring the distance between vectors. As it is said in section 2.1, an image is divided into a number of regions and as it is said in section 2.2 each region is represented by its average color and number of pixels. Hence the image as a whole is represented by many vectors, one vector for each region. Quadratic Form (QF) Distance, Mahalanobis Distance, and Kullback-Leibler (KL) Divergence and Jeffrey-Divergence (JD) are computationally complex. If the image were represented by few numbers of vectors, it might be possible to use these measures. To reduce the computational complexity, the Euclidean distance (Minkowski-Form Distance with p=2) is used as the similarity measure between the average colors of regions. In addition to average color, regions are compared by their number of pixels. The similarity measure is discussed in chapter 5.

2.4 Existing CBIR Systems

As discussed in chapter 1, section 1.3, existing CBIR systems can be grouped into two. In ***full-image*** retrieval systems features are extracted for the whole image without segmenting it into regions. In ***region-based*** systems before features are extracted the image is segmented into regions. Then features are extracted for each sub-part. ***Region-based*** systems can be further divided into three:

In the first type, the query image is not segmented but the images in the database are segmented and the system looks for images that contain the query image as their part. This is called sub-image retrieval. In the second type, both the query and the images in the database are segmented but only one part of the query image is used for the searching. In the third type both the query image and images in the database are segmented and all the regions of the query image will be used for the comparison.

Some of the existing systems are discussed below.

Blobworld [10]

Blobworld is developed by UC Berkeley Computer Vision Group. It segments the image into blobs (regions) using an EM-algorithm (Expectation-Maximization) based on the color and texture properties of the pixels. The user then selects interesting blobs and defines their relationship. Comparison is based on color histogram, mean texture contrast and anisotropy as feature descriptors of the blobs

The Earth Mover's Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval [30]

In this retrieval system, images are considered as points in a metric space in which they are moved around so as to locate image neighborhoods of interest, based on color information. The database images are mapped to distributions in color space, these distributions are appropriately compressed, and then the distances between all pairs I; J of images are computed based on the work needed to rearrange the mass in the compressed distribution representing I to that of J. When changing one signature to another, the work is the sum of the work done by moving the weights of the individual points of the source signature to those of the destination signature. This distance function is called the EMD. The system also uses of multi-dimensional scaling (MDS) techniques to embed a group of images as points in a two- or three-dimensional Euclidean space so that their distances are preserved as much as possible. Such geometric embeddings allow the user to perceive the dominant axes of variation in the displayed image group. In particular, displays of 2-d MDS embeddings can be used to organize and refine the results of a nearest-neighbor query in a perceptually intuitive way. By iterating this process, the user is able to quickly navigate to the portion of the image space of interest.

NETRA [31]

NETRA is developed by department of Electrical and Computer Engineering, University of California at Santa Barbara. Images are automatically segmented into about 6-12 non-overlapping homogeneous regions. The segmentation is based on an edge flow algorithm

which uses 'edges' in color and texture features to detect homogeneous regions. The user starts a search by selecting a category. He then selects an image and makes a query by selecting regions and features.

PicSOM [32]

PicSOM is developed by Laboratory of Computer and Information Science, Helsinki University of Technology. Image is divided into 5 regions. For each region, color and texture properties are used. In addition, edge and shape properties are used as features Features are stored in a tree structure that uses self-organizing map (SOM). The user starts by selecting one image and the search engine selects a preliminary set of similar images by comparing their tree-representation. Each unit in the tree has a weight that is equal for all units at the beginning. The user can then refine the search by marking the images with positive and negative values depending on whether the user finds them similar or not.

SIMPLIcity (Semantics-sensitive Integrated Matching for Picture LIbraries) [12]

SIMPLIcity is developed by James Ze Wang, Gio Wiederhold, Jia Li and others at Stanford University. It partitions the image into 4 x 4 pixel blocks and extracts a feature vector for each block. Use the k-mean clustering approach to segment the image into regions. The segmentation result is fed into a classifier that decides the semantic type of the image. An image is classified as one of the *n* pre-defined mutually exclusive and collectively exhaustive semantic classes. Features including color, texture, shape and location information are then extracted for each region in the image. The features selected depend on the semantic type of the image. The image is represented by the collection of features for all of its regions. The user simply clicks on an image or specifies an ID-number or URL and gets similar images.

Blobworld, Netra, Picsom and SIMPLIcity use segmentation to divide the image into few regions that are nearer to semantic objects. All of them use shape feature which needs the segmentation to be accurate. But it is impossible to get a segmentation technique that accurately segments all images into their semantic objects. Here lies the weakness of the

existing systems. The last system that uses EMD is discussed here for comparison purposes. It is a system that uses color feature alone and hence it is used as a comparison since the proposed system also uses color feature.

2.5 Image Databases

Rapid advances in computers and communication technology is pushing the existing information-processing tools to their limits. The past few years have seen an overwhelming accumulation of media-rich digital data such as images, video, and audio. The Internet is an excellent example of a distributed database containing several millions of images. Other examples of large image databases include satellite and medical imagery. [31]

The performance of the system is tested by using two image databases. The first image database is a subset of COREL image database and has 1000 images grouped into 10 semantically different categories as listed in table 2.2 below. These 1000 images are used by SIMPLIcity system to compare the performance of SIMPLIcity with that of the system developed by EMD [12].

Category No	**Category Name**
1	African people and village
2	Beach
3	Buildings
4	Buses
5	Dinosaurs
6	Elephants
7	Flowers
8	Horses
9	Mountains and glaciers
10	Food

Table 2.2 Semantic Category

Each semantic category includes images containing the specified objects. For example, the “Beach” category refers to scenery at coasts or river banks. “Flower” category contains images of flowers. Images in the flower category are shown in Table 2.3.

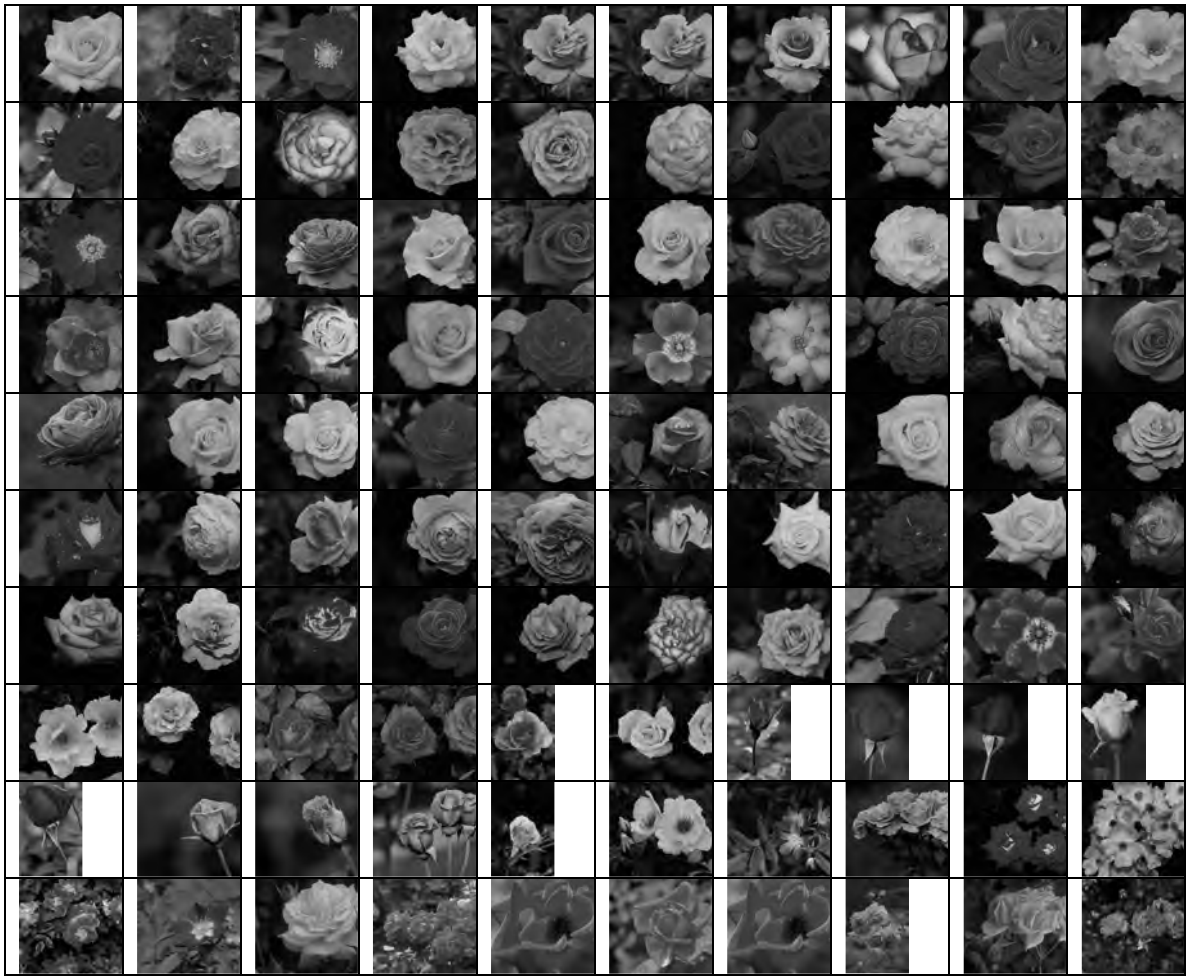


Table 2.3 Flower Category

The other image collection used in this work is the one used by WBIIS [33] and has 10,000 image collections. This image database is used to test the retrieval speed of the system. Both image collections were downloaded from www [34].

2.6 Conclusion

In this chapter, the different segmentation techniques, the different types of color features, the different types of similarity measures and some of the existing CBIR systems have been surveyed. In addition the image database used for testing the performance of the proposed system is also discussed. In each section, the best type is selected from among the different alternatives based on the survey. The conclusions drawn in each section are summarized as follows:

Since the system to be developed is not heavily dependent on the output of the segmentation, the less computationally expensive K-Means algorithm is used for the segmentation part. Since, almost all regions are used for the comparison, it is reasonable to use lower dimension color features like color moments. Hence, regions are represented by their average color content. In addition to the average color, number of pixels is also used to represent each region.

In this thesis, the K-Means segmentation algorithm, the first color moment (average color) and the Euclidean distance measure along with the number of pixels are used to measure the similarity of regions. The block diagram drawn in section 1.5 is redrawn here with the selected techniques incorporated.

The rest of the paper is organized as follows: chapter 3 discusses about the K-Means Segmentation algorithm, Chapter 4 discusses about the feature extraction and chapter 5 discusses about the proposed similarity algorithms based on the average color and number of pixels of regions. Performance measurement is discussed in chapter 6 and conclusion and future work is drawn in chapter 7.

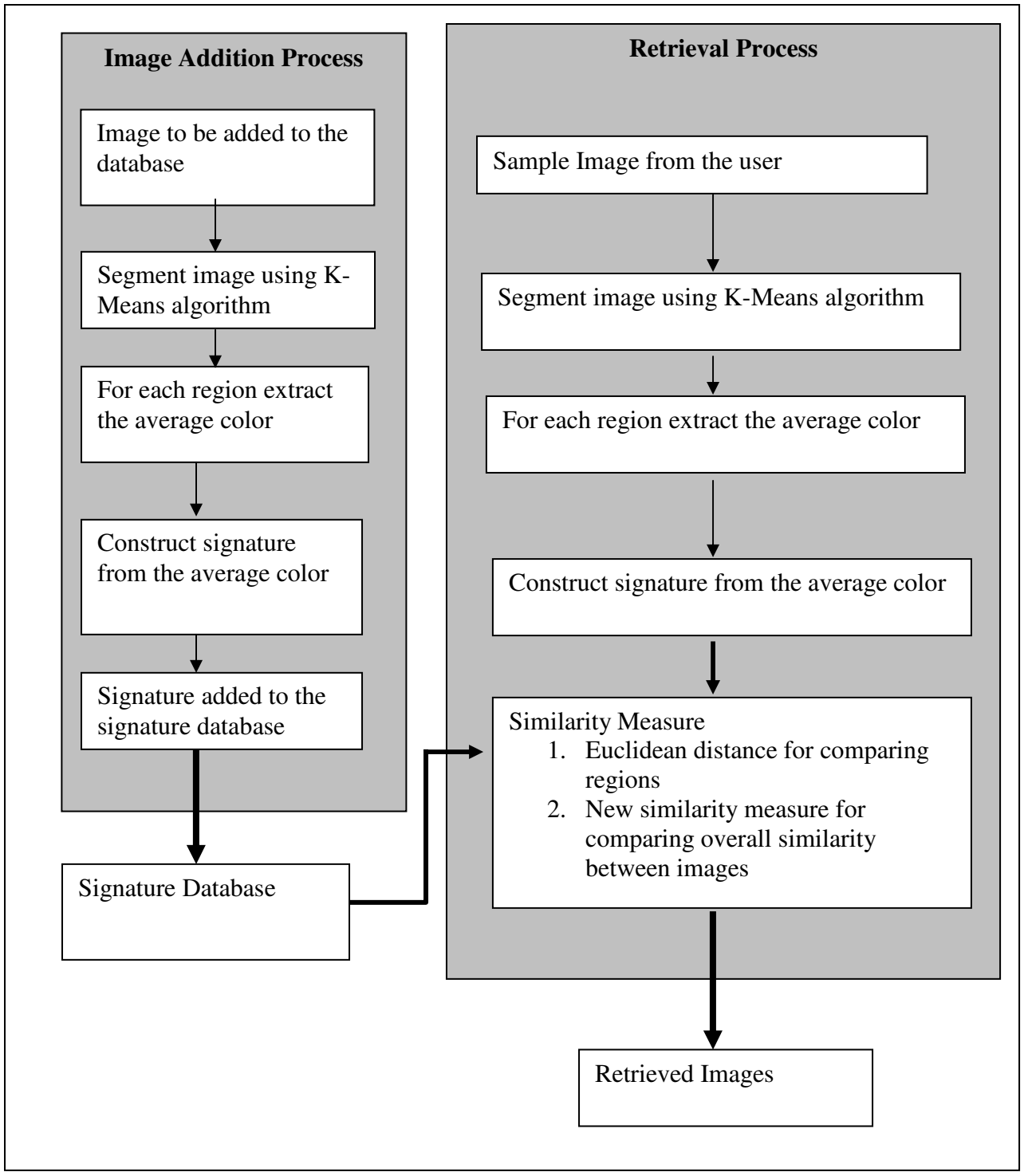


Figure 2.1 Block Diagram of the new system

Chapter 3

K-Means Segmentation Algorithm

3.1 Introduction

The term image segmentation refers to the partition of an image into a set of regions that cover it. The goal of image segmentation in many tasks is for the regions to represent meaningful areas of the image as shown in figure 3.1 below.

Figure 3.1. Image Segmentation Example

The main objective of segmentation is to decompose the image into parts for further analysis. In some cases, the environment might be well enough controlled so that the segmentation process reliably extracts only the parts that need to be analyzed further. An example is human face recognition.

In relation to CBIR systems, segmentation is used to acquire a region-based signature. Specially, to use shape feature, reliable segmentation is critical. Most of the current CBIR systems use shape feature and hence need a reliable segmentation.

As it is already stated in chapter 2, in section 2.1, there are a number of image segmentation techniques with their own advantages and disadvantages. One of the most popular techniques for image segmentation is the K-Means clustering. The main reason for the popularity of K-Means clustering segmentation is its speed advantage. Although it is fast it is not as refined as some recently developed methods.

In this work, the segmentation is needed for segmenting the image into regions which are far from representing a semantic object. So the less refined K-Means clustering segmentation algorithm is used here because of its speed advantage.

The chapter is organized as follows: section 3.2 discuses about the K-Means algorithm. Section 3.3 discusses about the experimental results.

3.2 Algorithm

Clustering in pattern recognition is the process of partitioning a set of patter vectors into subsets called clusters. There are a number of clustering algorithms used in different applications. Some of the clustering algorithms that are used for image segmentation include K-Means and Isodata clustering.

When clustering is used for image segmentation, the vectors represent pixels or sometimes small neighborhoods around pixels. The components of these vectors can include [35]

1 Intensity values

2 RGB values

3 Texture measurements

Any feature that can be associated with a pixel can be used to group pixels. In this work, RGB values are used to group pixels.

In traditional clustering, there are K clusters c_1, c_2… c_k with means m_1, m_2… m_k. A least square error measure can be defined as

$$D = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - m_k\|^2. \qquad 3.1$$

which measures how close the data are to their assigned clusters. A least-square clustering procedure could consider all possible partitions into K clusters and select the one that minimizes D. Since this is computationally infeasible, practically it is implemented as approximations by limiting the maximum number of iterations. One

important issue is whether or not K is known in advance. Many algorithms expect K as parameter from the user. Others attempt to find the best K according to some criterion.

Expecting K as a parameter from the user has a number of disadvantages. One disadvantage is that it creates a burden on the user. The other one is if the user selects different values at different times, inconsistency will be created leading to inaccurate results. Attempting to find the best K on some criterion is not considered here. It is left out for future work. In this work, a fixed value of 16 is used as cluster centers. In addition the maximum number of iteration is taken to be 30. These numbers are selected based on experimental results.

The algorithm is implemented in two steps. In step 1, the cluster to which a given pixel belongs is identified and given as an input to step 2. In step 2, based on cluster grouping, regions are identified. Neighboring pixels that belong to the same cluster form a region. Two options were tried to implement Step 2. The first option is called Region identification without merging and the second one is called Region identification with merging. The algorithm is discussed below.

Step 1: Cluster Identification

1. There are 16 cluster centers $k_1, k_2, \ldots, k_{16}$
2. For each pixel in the data, compute its distance d_i from the cluster centers using RGB value
3. Assign the pixel to the cluster whose distance is the smallest
4. Generate new cluster centers from the average of the pixels in a given cluster
5. Repeat steps 3 and 4 until maximum iteration is reached or there is no more change

At the end of this procedure, all pixels are grouped into one of the 16 clusters.

Assume that we have an image with total number of pixels equal to 56. Again assume that the image has a width of 14 pixels and height of 4 pixels. The number of cluster center selected for this example is four. Figure 3.2 shows the result of step 1 applied to an image with a fixed cluster number of 4. The 56 pixels in the image are grouped to one of the four clusters.

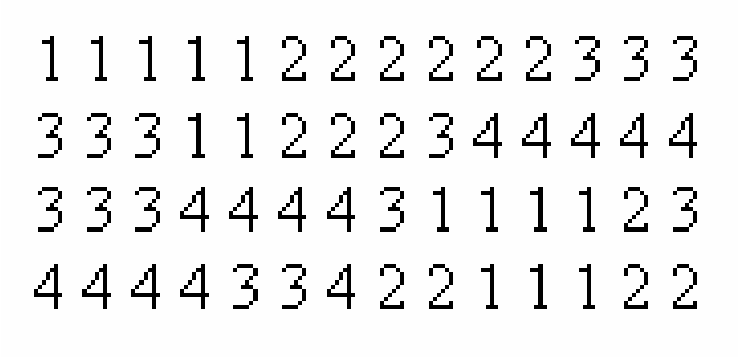


Figure 3.2. Output of Clustering

Step 2 Region Identification

Option 1 – without Merging

1. Initially all pixels will have a region number of -1.
2. Set regionCount = 0
3. Start from the pixel that is found at the upper left corner of the image and do the following
    a. If region number of the pixel is -1
        i. Start a new region by incrementing regionCount
        ii. Assign region number of the pixel to be regionCount
        iii. Find the neighbors of the pixel
        iv. If the region number of the neighboring pixel is equal to -1 and the cluster number is equal to the cluster number of the current pixel, then assign the region number of the neighboring pixel equal to that of the current pixel.
    b. Else
        i. Find the neighbors of the pixel
        ii. If the region number of the neighboring pixel is equal to -1 and the cluster number is equal to the cluster number of the current pixel, then assign the region number of the neighboring pixel equal to that of the current pixel.
4. Repeat step three for all pixels

Option 2 – with Merging

1. Initially all pixels will have a region number of -1.
2. Set regionCount = 0
3. Start from the pixel that is found at the upper left corner of the image and do the following
    a. If region number of the pixel is -1
        i. Start a new region by incrementing regionCount
        ii. Assign region number of the pixel to be regionCount
        iii. Find the neighbors of the pixel
        iv. If the cluster number of the neighboring pixel is equal to the cluster number of the pixel, then
            1. If region number of the neighboring pixel is equal to -1 then assign the region number of the neighboring pixel equal to that of the current pixel.
            2. else merge the two regions in which the neighboring pixel and the current pixels are found
    b. Else
        i. Find the neighbors of the pixel
        ii. If the cluster number of the neighboring pixel is equal to the cluster number of the current pixel, then do one of the following
            1. If region number of the neighboring pixel is equal to -1 then assign the region number of the neighboring pixel equal to that of the current pixel.
            2. else merge the two regions in which the neighboring pixel and the current pixels are found
4. Repeat step three for all pixels

In the second step, neighboring pixels that belong to the same cluster form a region. A pixel at row i and column j will have four neighboring pixels with coordinates (i, j+1), (i+1, j+1), (i+1, j), and (i+1, j-1). Figure 3.3 shows this for a pixel at position (3, 4). Pixels that are found at the edges will have fewer numbers of neighboring pixels.

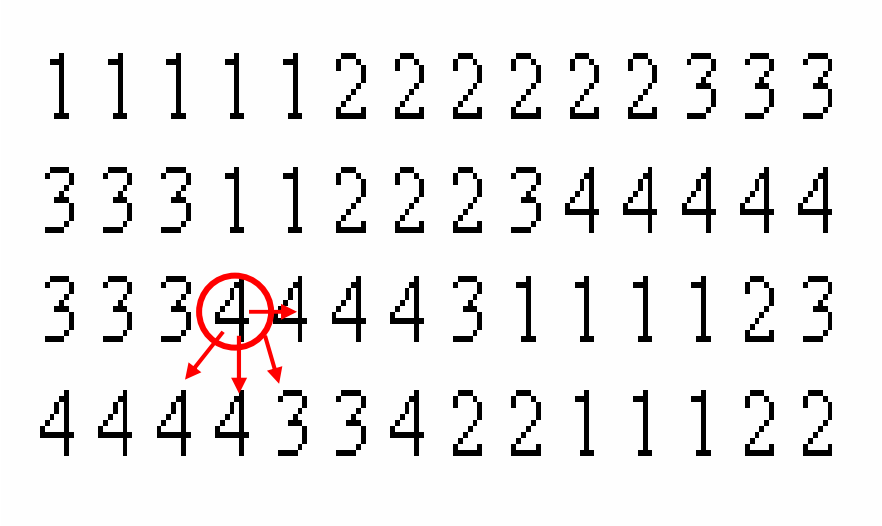


Figure 3.3. Neighbors of a pixel

The algorithm starts from the top-left pixel and continues processing the pixels row-wise. Assume we have reached pixel (3, 4), processing all the pixels before it and identifying the corresponding regions. Pixels that are found before (3,4) and that can take (3,4) as their neighbor pixel are (2,3), (2,4), (2,5) and (3,3). All of these pixels have a different cluster group to that of (3, 4). Hence (3, 4) is not part of any region yet and a new region will start at pixel (3, 4).

Pixel (3, 4) has four neighbors: (3, 5), (4, 5), (4, 4) and (4, 3). From these neighbors, (3, 5), (4, 4) and (4, 3) have the same cluster as that of (3, 4). These three neighboring pixels will be part of the new region that is started at pixel (3, 4). The next pixel to be processed is pixel (3, 5). A new region will not be started at pixel (3, 5) because it is already part of the region started at (3, 4). Neighbors of (3, 5) are (3, 6), (4, 6), (4, 5) and (4, 4). Only (3, 6) and (4, 4) have similar cluster group to that of (3, 5). (4, 4) is already part of the region. (3, 6) will be added to the region. By the same token (3, 7) and (4, 7) will be part of the region started at (3, 4). So, the region that started at (3, 4) will include pixels (3, 5), (3, 6), (3, 7), (4, 7), (4, 4) and (4, 3).

Assume we have reached at pixel (4, 1) continuing the process. Pixel (4, 1) starts a new region because the neighbors before it belong to different cluster centers. It has only one neighbor, pixel (4, 2). Since the two have the same cluster center, (4, 2) will be part of the region started at pixel (4, 1). When we reach at pixel (4, 2) a new region will not be

started because it is part of the region started at pixel (4, 1). The neighbor of pixel (4, 2) is pixel (4, 3). As we have seen above, pixel (4, 3) is already part of the region started at (3, 4). Now, comes the difference between the two options. In the case of option 1 the two regions remain as separate regions. This is shown in figure 3.4. In option two, the two regions will be merged into one region. This is shown in figure 3.5.

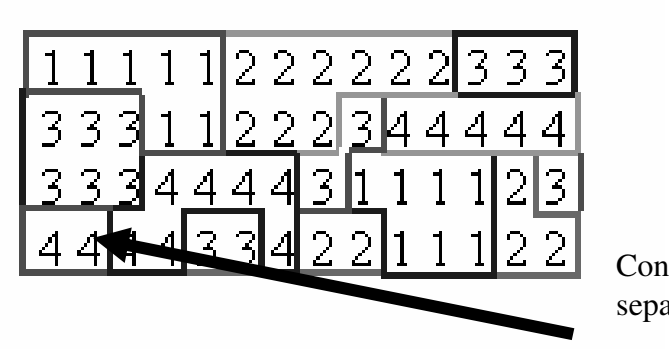


Figure 3.4 Segmentation without merging

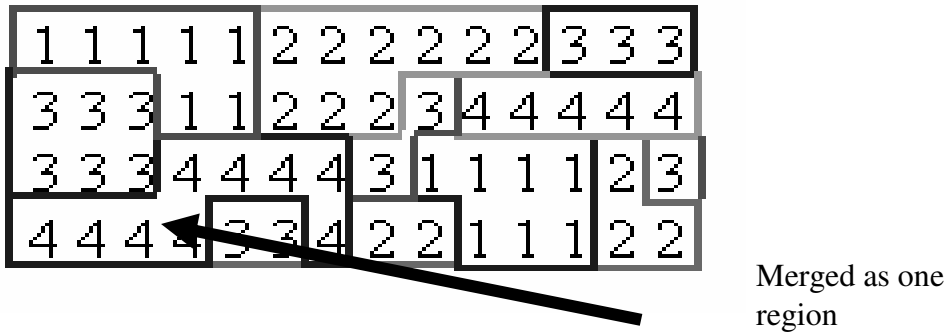


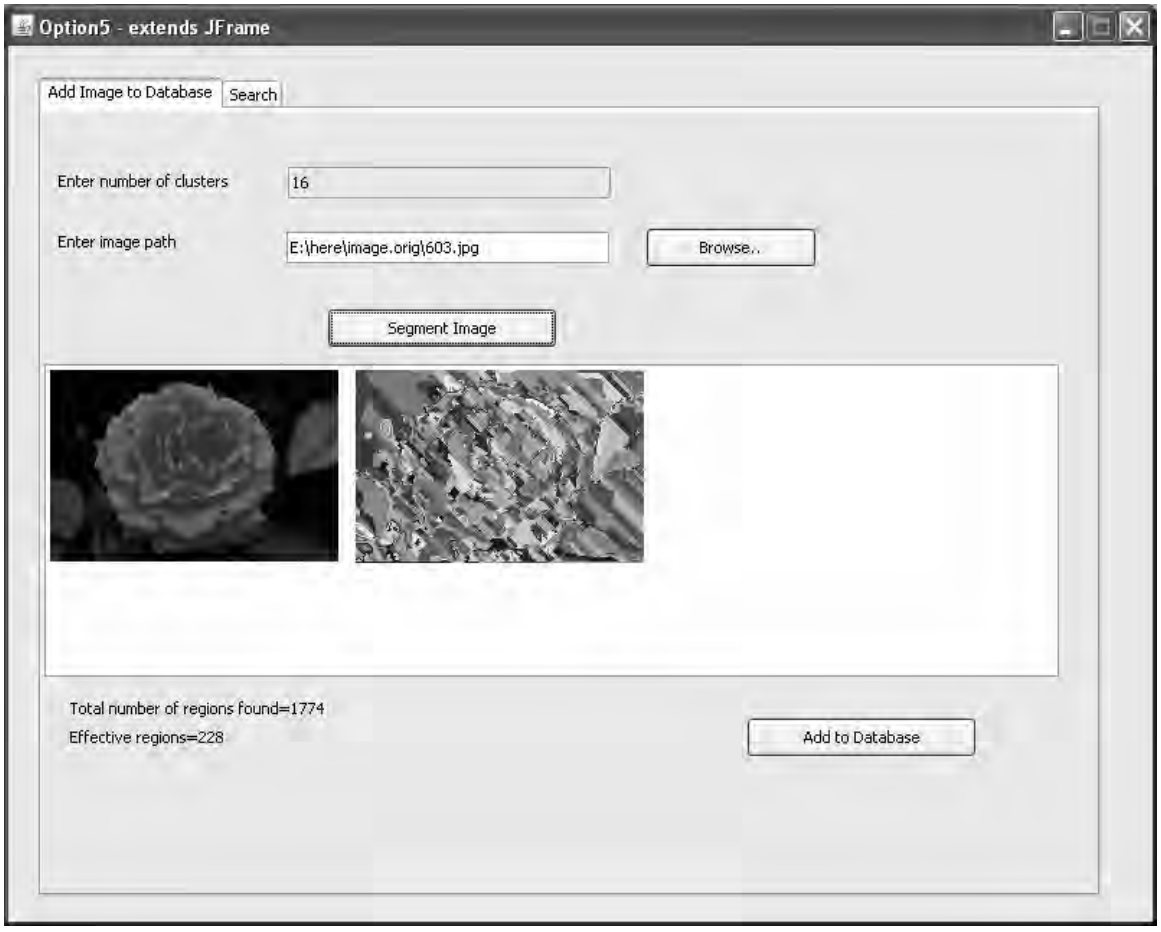
Figure 3.5 Segmentation with merging

The advantage of the first option is its speed. The advantage of the second option is that it decreases the number of regions. In this work, the second option is used to reduce the number of regions. This reduces the time for searching in the image database. If the image is represented by more regions, it will increase the searching time.

3.3 Experimental Results

Figure 3.6 shows the output of option 1 when applied for the rose image from the flower category. As we can see, in the output, the rose image is divided into 1774 regions. Most of these regions are very small and difficult to locate in the output. These small regions

will not be considered in the searching. Regions whose number of pixels is less than 0.1% of the total number of pixels of the image will not be used in the searching and will be left out. The regions whose size are greater or equal to 0.1% of the size of the image are called ***effective regions.*** Effective regions are used in the searching. Of the 1774 regions, only 228 are effective regions.



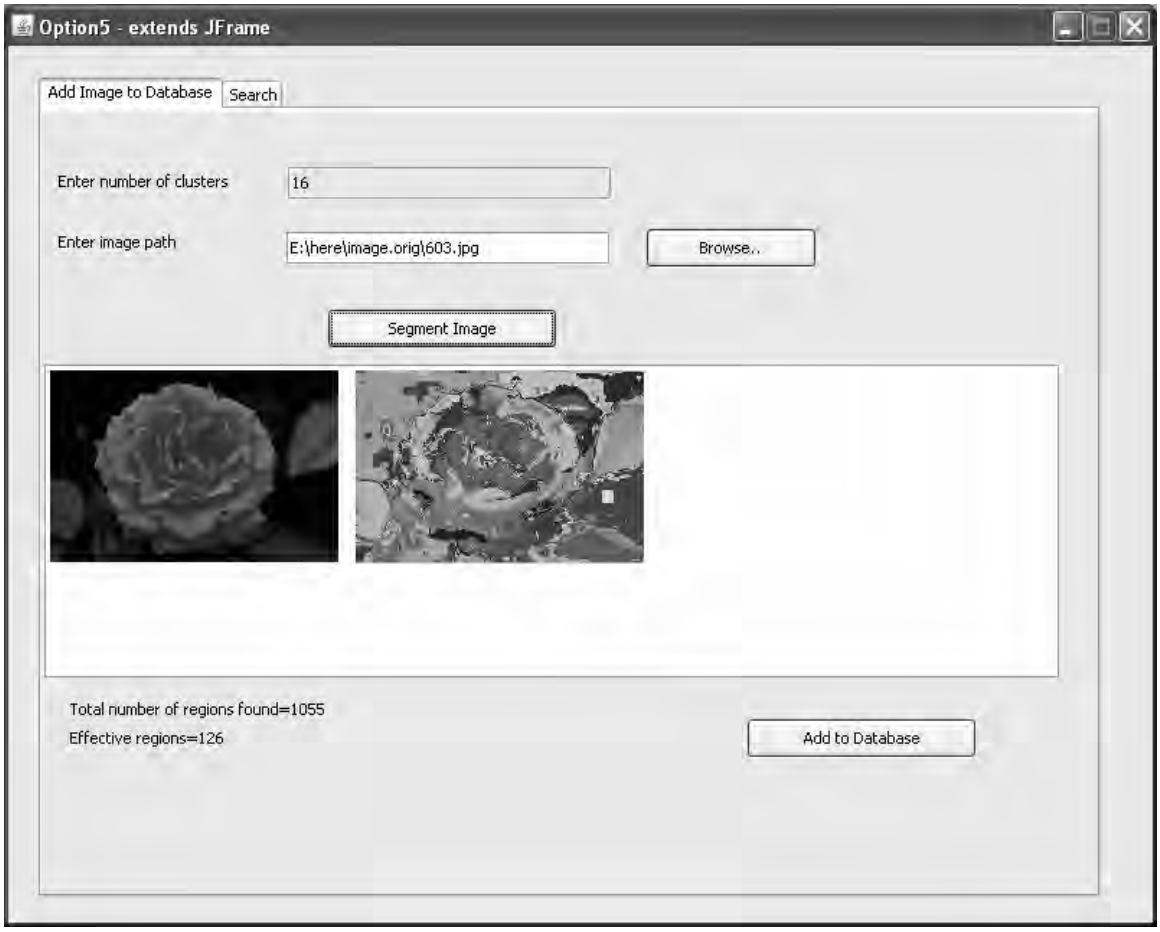
Segmentation without merging

Total number of regions 1774

Effective regions 228

Figure 3.6 Option 1 output

In the case of option 2, the total numbers of regions are 1055 and the effective regions are 126 as shown in fig 3.7. As the second option decreases the number of regions, it is the one that is used in this thesis work.



Segmentation with merging

Total number of regions 1055

Effective regions 126

Figure 3.7 Option 2 Output

As can be seen in the results above, since the segmentation is unreliable, the regions identified are far from representing an object and the number of regions is too high. Since it is not feasible to work with all the identified regions, only regions that represent 0.1% of the total size of the image will be used for the searching.

The GUI, used for segmenting the images, is discussed in Appendix A. The source code for the segmentation is shown in Appendix D1.

Chapter 4

Feature Extraction

4.1 Introduction

As it is already stated, in section 2.2, CBIR uses the visual contents of an image such as color, texture and shape to convert it to some mathematical form and store it in a feature database. The mathematical form, which is called the signature of the image, is usually described by multi-dimensional feature vectors. The feature vectors of the images form a feature database.

The performance of CBIR system depends on the features used to describe the contents of the image. Color features are among the most important and extensively used low-level features in image database retrieval. Combined with image segmentation, color features can be used to describe the content of an image.

As it is stated in chapter 2, section 2.2, the widely used representations for color feature include color moments and color histogram. In this thesis work, average color and number of pixels of a region are used to represent each region of an image in the feature database.

The chapter is organized as follows: section 4.2 discusses about the color moment and the experimental results are discussed in section 4.3.

4.2 Color Moment

Color moments are measures that can be used to differentiate images based on their features of color. Once calculated, these moments provide a measurement for color similarity between images. These values of similarity can then be compared to the values of images indexed in a database for tasks like image retrieval.

The basis of color moments lays in the assumption that the distribution of color in an image can be interpreted as a probability distribution. Probability distributions are characterized by a number of unique moments (e.g. Normal distributions are

differentiated by their mean and variance). It therefore follows that if the color in an image follows a certain probability distribution, the moments of that distribution can then be used as features to identify that image based on color.

Stricker and Orengo [23] used three central moments of an image's color distribution. They are Mean, Standard deviation and Skewness. A color can be defined by 3 or more channels. (Here we will restrict ourselves to the RGB color space). These three moments are discussed in chapter 2 in section 2.2.2.

As we have seen in chapter three, the output of the segmentation is dividing the image into multiple regions. Because the number of regions is too high, each region should be represented by a feature vector whose dimension is very low. Since there are too many regions, the variation of the color content with a given region is very small. Hence we can represent each region by its average color.

Defining the i^{th} color channel at the j^{th} image pixel as p_{ij}, the formula for the average color of a region having N pixels will be as follows.

$$E_i = \sum_{N}^{j=1} \frac{1}{N} p_{ij} \qquad 4.1$$

Since there are three color channels (i.e. red, green and blue), the average color of a region is represented by three numbers.

In addition to the average color of a region, the number of pixels of the region is also stored as part of the feature vector. The number of pixels of a region is used in the similarity algorithm as it is discussed in the next chapter.

4.3 Experimental results

Sample feature data for the rose image of Figure 3.7 is shown in Table 4.1. In this table, an image is represented as a collection of regions, each region being represented by its average color and number of pixels, as discussed above. The average color is calculated in RGB space for the three color channels of red, green and blue. Although the large

number of effective regions increases the searching and comparison time, the lower dimension of the feature vector by which each region is represented reduces the computational complexity. Additional reduction in complexity can be obtained by using similarity measures of low complexity.

Region No	**Red**	**Green**	**blue**	**noOfPixels**
1	2	2	2	96
2	2	2	1	41
3	7	7	7	67
4	3	2	1	73
5	7	7	7	553
6	201	91	89	30
7	3	3	1	195
8	208	23	13	54
9	203	19	15	90
10	13	4	5	30

Table 4.1 Sample Feature Table

The complete feature table for the sample image and the GUI used for generating the feature database are discussed in Appendix B. The source code for feature extraction is shown in Appendix D.2.

Chapter 5

Similarity Measure

5.1 Introduction

In the previous chapter we have seen the procedure to extract the color features of an image and store it in a feature database. Once the feature database is constructed, the next step is to have a measure of similarity to compare the features of the query image with that of the images in the database. Based on the calculated visual similarities between a query image and images in the database, list of images, ranked by their similarities with the query image are returned as retrieval result.

The proposed system is a region-based system that uses unreliable segmentation to divide an image into a number of regions that are far from representing semantic objects. As discussed in chapter 4, section 4.2, regions are represented by their average color and by their number of pixels.

The proposed system differs from the existing systems in two ways

1. It uses unreliable segmentation
2. It uses new similarity measure to compare the similarity of images

The segmentation technique is discussed in chapter 3. In this chapter the similarity measure is discussed.

The chapter is organized as follows: the similarity algorithm is discussed in section 5.2 and section 5.3 discusses about the experimental results. Finally, conclusion is drawn in section 5.4.

5.2 Similarity Algorithm

In this section, the new similarity algorithm is discussed. The proposed similarity algorithm tries to measure the similarity of two images based on the similarity of regions of the images. If two images have more number of similar regions, it is likely that the two images have similar visual content. The similarity algorithm can be divided into two steps: region similarity and overall similarity. Region similarity is discussed in section 5.2.1 and overall similarity is discussed in section 5.2.2

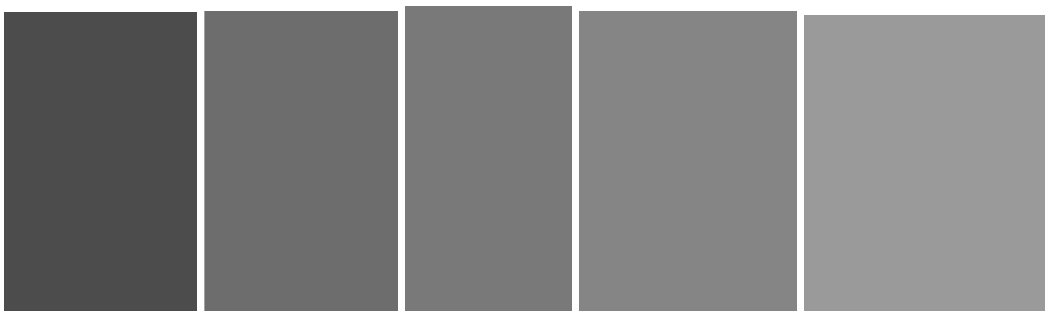
5.2.1 Region Similarity

The similarity of two regions is measured using their color content and number of pixels of the two regions. Color content of a region is expressed by its average color as discussed in chapter 4, section 4.2. Similarity of average colors is measured by using the Euclidean distance. Here average colors are assumed to be points in three dimensional space. Points in three dimensional space are considered nearer to each other if the Euclidean distance between them is small. Hence, smaller the Euclidean distance, more similar the color content of the two regions. Identical colors will have a Euclidean distance of zero.

Assume that we have two regions, R1 and R2 with color content of (r1, g1, b1) and (r2, g2, b2) respectively. The Euclidean distance between the two regions is expressed as follows as shown in equation 5.1.

$$D = \sqrt{(r1 - r2)^2 + (g1 - g2)^2 + (b1 - b2)^2} \qquad 5.1$$

On a scale that ranges from 0 to 255 for each channel in the RGB color space, we can have a total number of 16,777,216 colors. But humans can identify the difference of very few combinations. For example consider the rectangles in figure 5.1. Even if the color content is different in the RGB combination, only the first and the last can be differentiated by humans.



255, 0, 0 255, 48, 48 255, 64, 64 255, 80, 80 255,112,112

Figure 5.1 Color Visualization

The Euclidean distance between the first and the last image is 158.3919. To increase the discriminating power of the distance, a low value of 20 is taken as a threshold, based on experimental findings. Two regions are said to have similar color content if the Euclidean distance between the average colors of the two regions is less than or equal to 20.

Comparing regions by their color content alone is not sufficient. If only color content is used as a measure, a dot can be considered similar to a region of larger size. To incorporate the size of regions in the similarity comparison, the number of pixels of each region is also taken as a measure. Here also, it is good if regions have sizes of comparable amount, instead of expecting them to have equal number of pixels. 1.5 is taken as a threshold ratio for two regions to be comparable in their sizes after a number of trials. Assume the number of pixels in region 1 is p1 and that of region 2 is p2. Assuming that p1 > p2, the condition can be expressed as in equation 5.2

$$p1 / p2 <= 1.5 \qquad 5.2$$

The algorithm for region similarity is summarized below.

Algorithm – Region Similarity

Two regions are said to be similar if the following two conditions are satisfied

1. the Euclidean distance between the average colors of the two regions should be less than or equal to 20 as given by equation 5.1

2. The ratio of pixels in the two regions should be less or equal to 1.5 as given by equation 5.2

5.2.2 Image Similarity

Once the similarity between regions is defined, the next step is to define the similarity of the two images as a whole based on the similarity of their regions.

In this work, the similarity of the images is expressed in two ways:

Option 1 is based on the sum of pixels in the similar regions and

Option 2 is based on the number of similar regions.

The two options are discussed below using a simple example.

Assume that we have two images, M1 and M2 in the feature database. Assume that M1 has 6 regions, and M2 has three regions. Assume that the query image, S has 5 regions. Table 5.1 shows the comparison of the regions of images in the database to that of the regions of the query image. ✗ indicates that the two regions are dissimilar whereas ✓ indicates that the two regions are similar. For example M11 is similar to S1. That means, the two conditions mentioned in the region similarity algorithm are met. Based on Table 5.1, M1 has 4 regions similar to the regions in the query image whereas, M2 has 3 regions similar to the regions in the query image. Based on the first option in which the similarity is based on the number of similar region, M1 is more similar to S than M2.

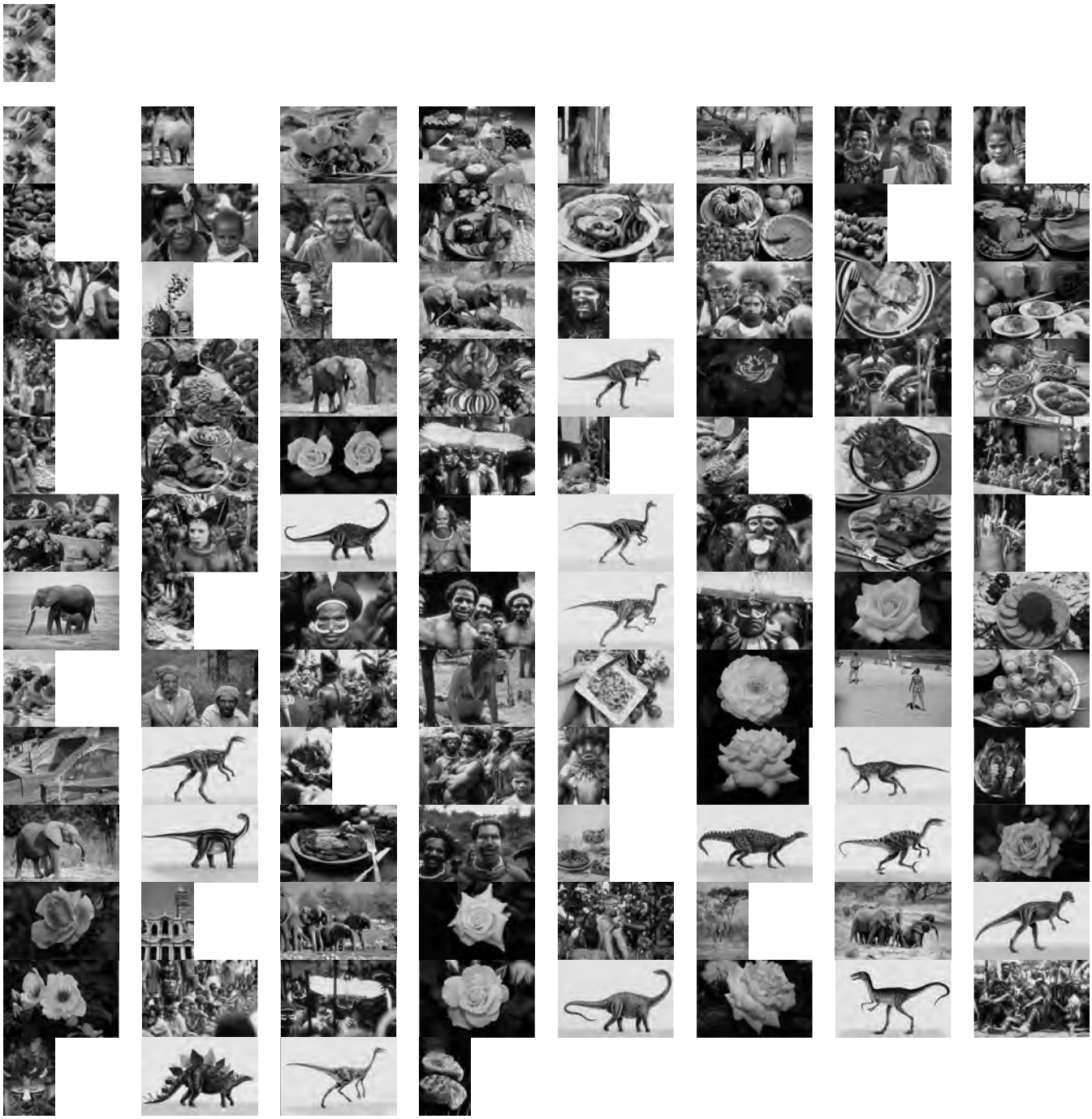
Assume that the number of pixels in S1, S2, S3, S4 and S5 is respectively 30, 20, 90, 60 and 10. The only region in the sample that has no similarity to regions of M1 is S3. So the sum of similar pixels for M1 becomes 30 + 20 + 60 + 10 = 120. For M2, the similar

regions are S1, S3 and S4. The sum of similar pixels becomes 30 + 90 + 60 = 180. So based on the second option, M2 is more similar to S than M1.

Image	Regions	Sample Image				
		S1 (30)	S2 (20)	S3 (90)	S4 (60)	S5 (10)
M1	M11	✓	×	×	×	×
	M12	×	×	×	×	✓
	M13	×	✓	×	×	×
	M14	×	×	×	×	×
	M15	×	×	×	✓	×
	M16	×	×	×	×	×
M2	M21	×	×	✓	×	×
	M22	✓	×	×	✓	×
	M23	×	×	×	×	×

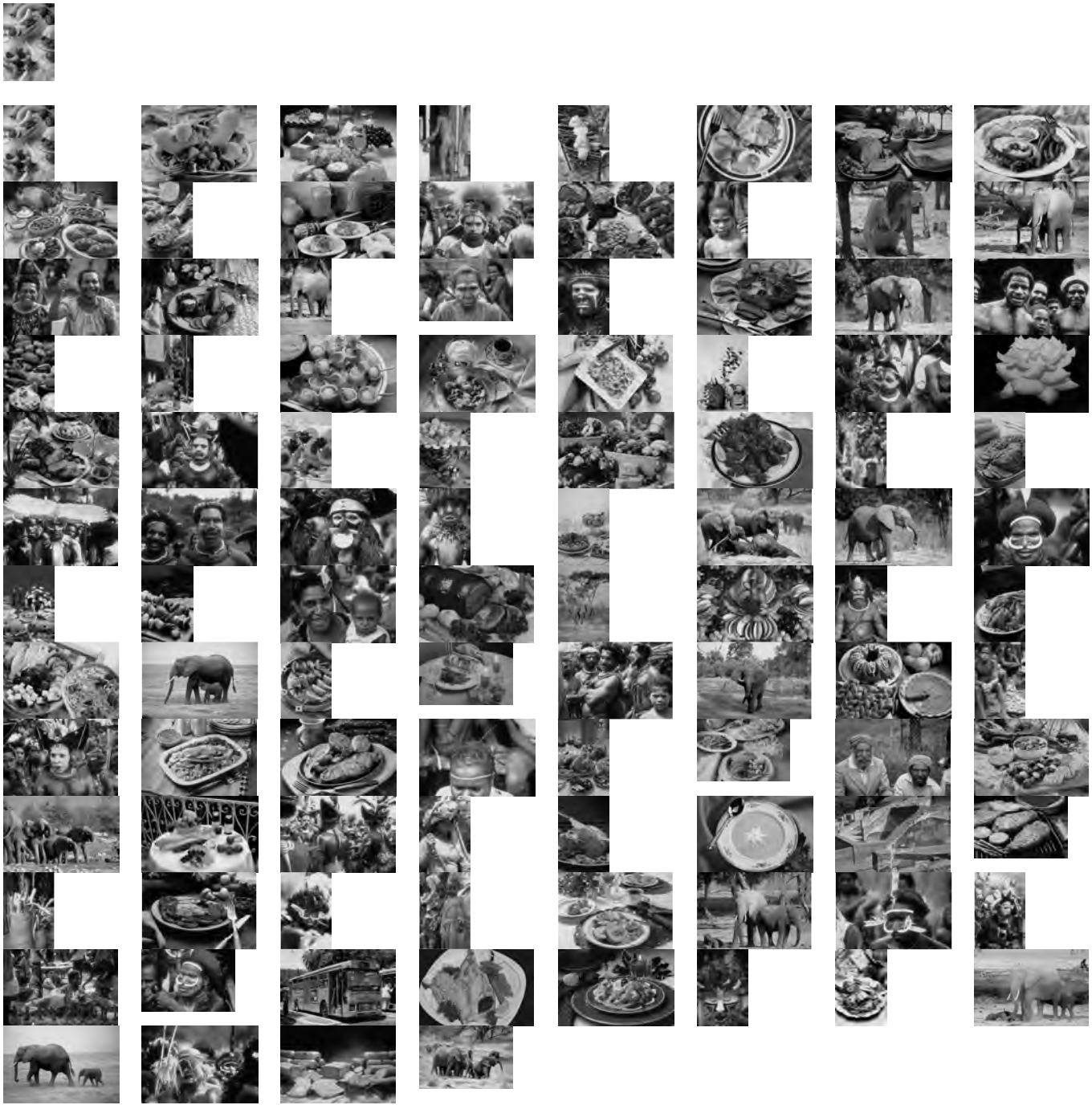
Table 5.1 Region Similarity

As can be seen in Table 5.1, a region in the query image is similar to at most one region of the image in the database. Similarly, a region in the image in the database is at most similar to one region of the query image. For example if M11 is similar to both S1 and S5 only one of the two will be considered as a match. This decision was reached based on experimental results as shown in the sample results in Table 5.2 and 5.3. The number of matches for the case of multiple region matching is by far lower than that of one-to-one region matching.



30 matches out of 100

Table 5.2 Multiple Region Image Search



50 matches out of 100

Table 5.3 One-to-One Region Image Search

The algorithms for the similar region count and pixel sum are presented below.

Option 1 – to find the sum of pixels in the similar regions of two images X and Y

1. set sum = 0
2. for each region K in X
    a. for each region M in Y
        i. if (M is similar to K and M is not matched before)
            1. add number of pixels of K to sum
            2. break
        ii. end if
    b. Next M
3. Next K
4. return sum

Option 2 – to count the Number of Similar Regions between two images X and Y

1. set counter = 0
2. for each region K in X
    a. for each region M in Y
        iii. if (M is similar to K and M is not matched before)
            1. increment counter
            2. break
        iv. end if
    b. Next M
3. Next K
4. return counter

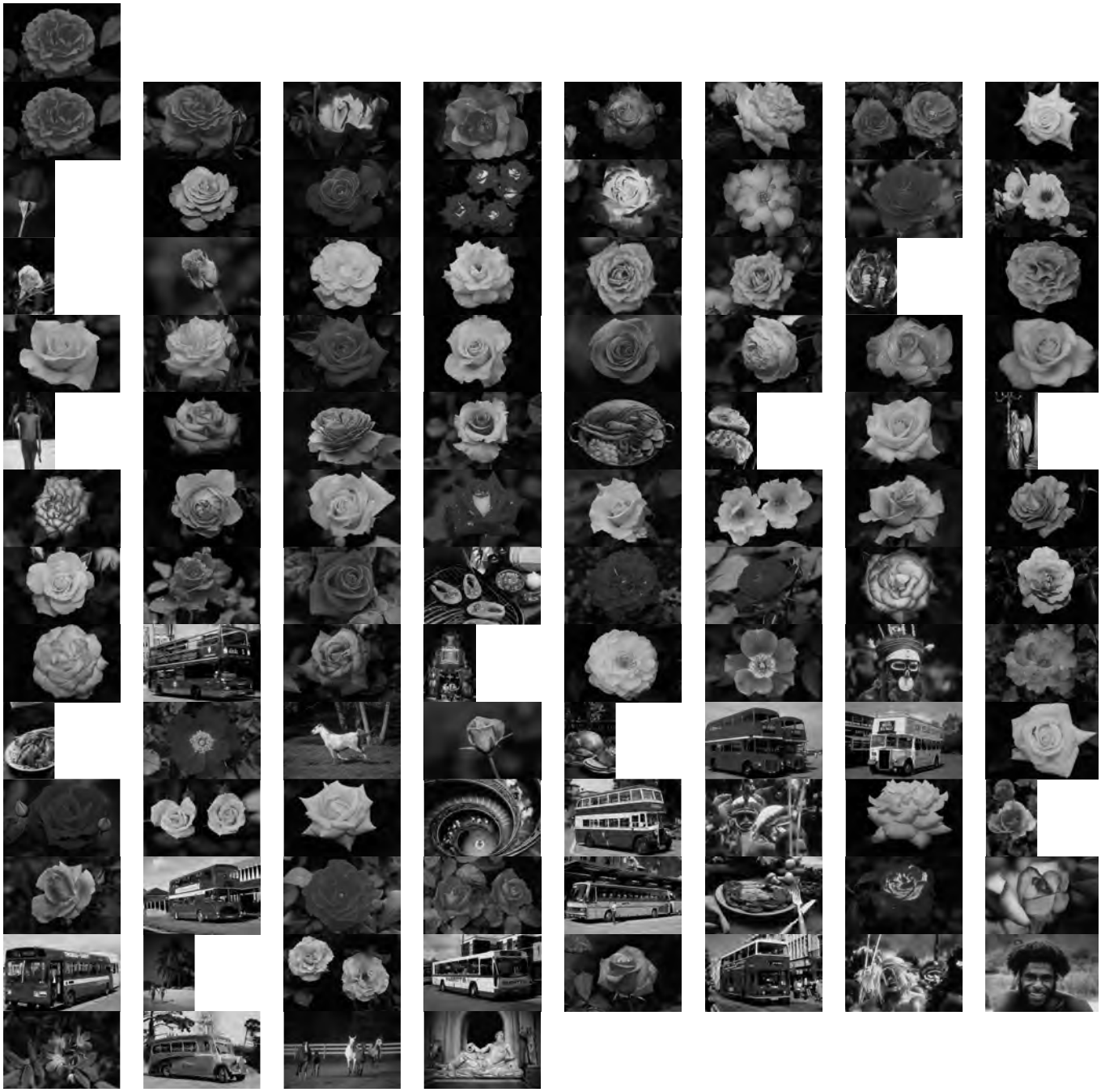
Note that when we say region M is similar to region K, it means that M and K satisfy the condition specified in the region similarity algorithm. Option 1 differs from option 2 by the use of the sum of number of pixels in the similar regions instead of counting the similar regions.

5.3 Experimental Results

As discussed in section 2.4, the proposed system is tested using two image collections. In this section, accuracy results from the 1000 image collection are shown. The sample outputs for the rose image from the flower category are shown below. Table 5.3 and Table 5.4 show sample results for the two options discussed above, i.e. pixel sum (option 1) and region count (option 2). Below the sample result, a match record is shown. An image in the result is counted as a match if it is in the same semantic category of the sample image. The sample image is the one shown at the first row of the tables. The outputs would be similar to Table 2.2 if the system were idle. All the images in the tables would be flowers. The outputs show the top 100 images returned by the system.

Of the 100 images returned by the system, 71 are flowers for Option 1 and 67 are flowers for option 2. Option 1 performs better for this sample. It is not possible to say option 1 is better than option 2 based on a single output. Comparison of the two options is discussed in chapter 6.

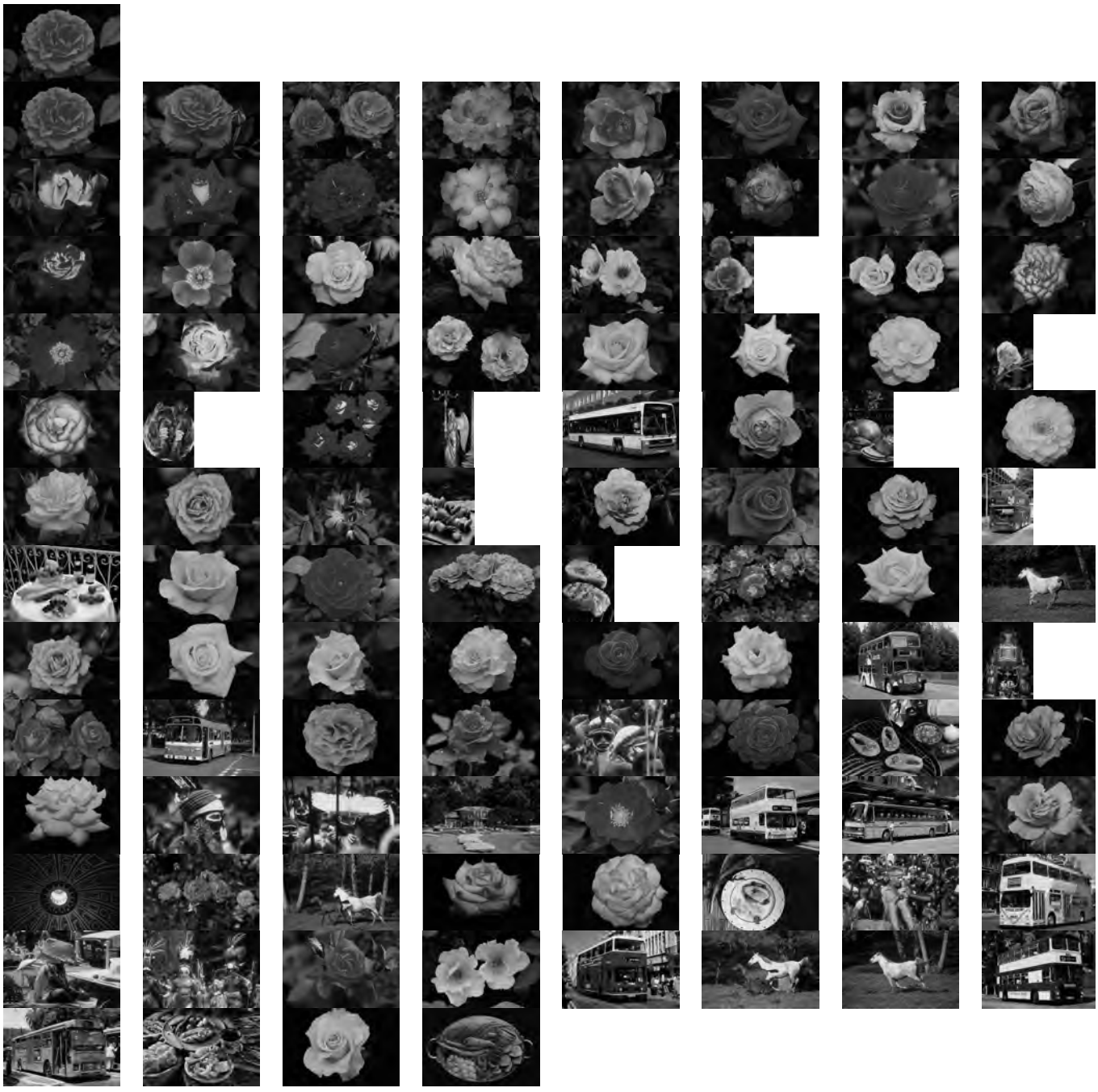
Pixel Sum - Option 1



71 Matches out of 100

Table 5.4 Sample output for Option 1 Similarity Algorithm

Region Count - Option 2



67 Matches out of 100

Table 5.5 Sample output for Option 2 Similarity Algorithm

5.4 Conclusion

In this chapter the proposed image similarity measure is discussed. Image similarity is based on one of the following options: either on number of count of similar regions or on sum of pixels of the similar regions. Similarity of regions is measured by the Euclidean distance of the average color and the number of pixels of the regions.

A region can be matched at most to one region. Multiple region matching is discarded based on experimental findings

The performance of the two image similarity options is compared in Chapter 6.

The GUI for image search is shown in Appendix C. The source code for similarity is shown in Appendix D.3 and D.4.

Chapter 6

Performance Metrics

6.1 Introduction

As it is said in chapter 5, section 5.3, the accuracy of the system is tested using an image collection of 1000 images. As it is said in chapter 2 section 2.5, the 1000 images are grouped into 10 categories, each containing 100 images, based on their semantic type.

In this chapter the performance of the proposed system is measured using three statistics. The three statistics, precision, average rank and standard deviation of the ranks are discussed in section 6.2. The 1000 image collection and the three statistics were used by SIMPLIcity system to measure its performance.

The other test discussed in this chapter is the speed test. The speed test measures how fast the system is. The speed test is done on an image collection of 10,000 images. This image collection is used by WBIIS.

The chapter is organized as follows: section 6.2 discusses about the different statistics that are used to measure the performance of CBIR systems.6.3 discusses about the accuracy test and 6.4 discusses about the speed test.

6.2 Metrics

CBIR is essentially an information retrieval problem. Therefore, evaluation metrics have been quite naturally adopted from information retrieval research. Two of the most popular evaluation measures are:

—Precision: The percentage of retrieved pictures that are relevant to the query.

—Recall: The percentage of all the relevant pictures in the search database which are retrieved.

Let us see numerical example to clarify the difference between precision and recall.

Consider the following data

Total number of images in the database	= 100
Total number of images relevant to the query image	= 16
Total number of retrieved images	= 20
Number of relevant images that are retrieved	= 10

Based on the above data

Precision = (Number of relevant images that are retrieved)/ (Total number of retrieved images)

= 10/20 = 0.5

Recall = (Number of relevant images that are retrieved)/ (Total number of images relevant to the query image)

= 10/16 = 0.625

If the system were ideal, both precision and recall will have a value of 1. Notice that when the query in question is a picture, relevance is extremely subjective.

These measures have a drawback as discussed in [4]. Usually CBIR systems use modified evaluation measures to test the performance of a system. In this thesis work, the metric used to measure the performance of the system is the one that is used by SIMPLIcity. The SIMPLIcity system was evaluated based on a subset of the COREL database, formed by 10 image categories (shown in Table 5.2), each containing 100 pictures. Within this database, it is known whether any two images are of the same category. In particular, a retrieved image is considered a match if and only if it is in the same category as the query. This assumption is reasonable since the 10 categories were chosen so that each depicts a distinct semantic topic. Every image in the database was tested as a query and the retrieval ranks of all the rest images were recorded. Three statistics were computed for each query:

1) The precision within the first 100 retrieved images,

2) The mean rank of all the matched images (the order in which it is retrieved), and

3) The standard deviation of the ranks of matched images.

Precision is the percentage of relevant images that are retrieved. The number of retrieved images is fixed to be 100. For this experiment, the total number of images, the total number of retrieved images and the total number of relevant images is as shown below

Total number of images in the database	= 1000
Total number of images relevant to the query image	= 100
Total number of retrieved images	= 100

Since total number of images relevant to the query image is equal to total number of retrieved images, recall and precision are identical for this test. If the number of relevant images that are retrieved is n, then both precision and recall will be n/100.

Mean rank is calculated by considering the rank of all the relevant images. All the 1000 images in the database will be ranked based on their similarity measure to the query image. Mean rank is the average of the rank of all relevant images.

The third parameter is the standard deviation of the ranks of the relevant images. This parameter measures the average distance of the ranks of the relevant images from the mean rank. The formula for the mean rank is given below

$$S=\frac{1}{N}\sqrt{\sum_{i=1}^{100}(ri-rm)2} \qquad 6.1$$

Where N is total number of images in the category, r_i is the rank of relevant image i, and r_m is the mean rank.

For a system that performs well, the value of precision should be high and those of average rank and standard deviation should be low.

The following subsections discuss the output of the comparison. Subsection 6.2.1 discusses the comparison between Option 1 and Option 2 similarity measures. Subsection 6.2.2 discusses the comparison between Option 2 and EMD and the comparison between Simplicity and Option 3 is discussed in subsection 6.2.3.

6.3 Accuracy Tests

6.3.1 Comparison of Option 1 and Option 2

As discussed in chapter 5, the proposed similarity measure has two options. In this section the options are compared using the three evaluation measures discussed above. The following graphs show the comparison for the three measurements.

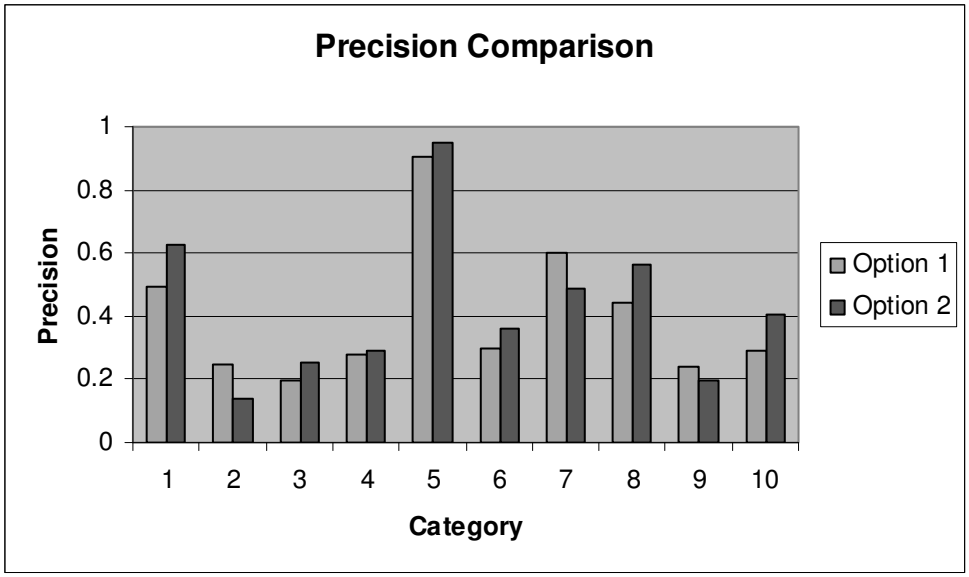


Figure 6.1 Precision Comparison between Option 1 and Option 2

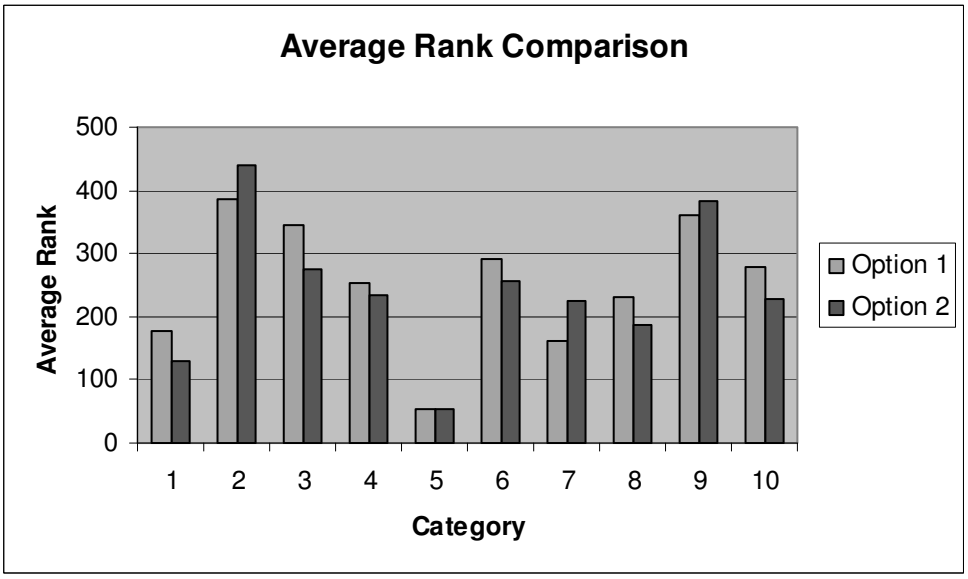


Figure 6.2 Average Rank Comparison between Option 1 and Option 2

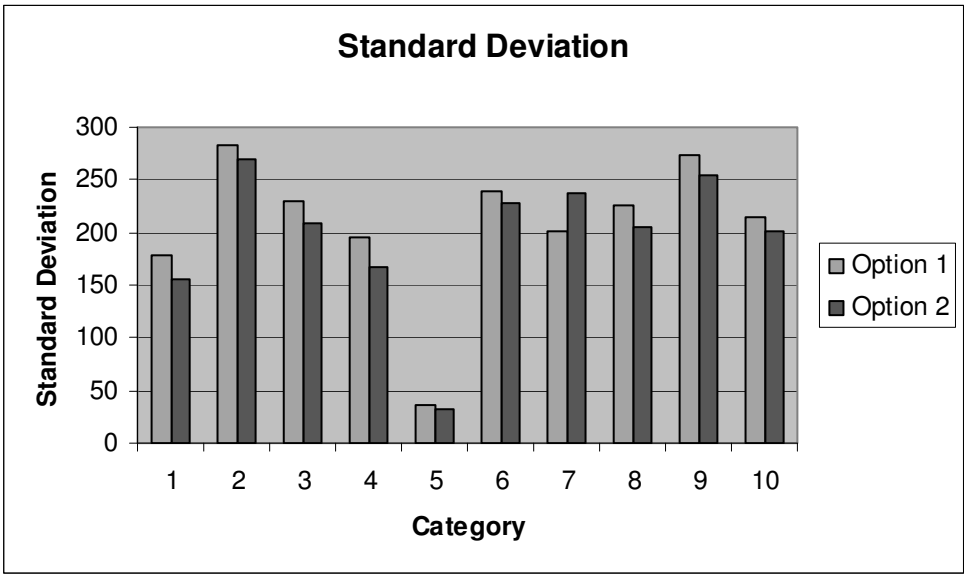


Figure 6.3 Standard Deviation Comparison between Option 1 and Option 2

System Name	Precision	Average Rank	Standard Deviation
Option 1 (Pixel Sum)	3 categories	3 categories	1 categories
Option 2 (Region Count)	7 categories	7 categories	9 categories

Table 6.1 Summary of Comparison of Option 1 and Option 2

As summarized in Table 6.1, Option 2 is better than option 1. Hence region count is better than pixel sum. The next subsections discuss the comparison between Option 2 with EMD and Simplicity.

6.3.2 Comparison with EMD

The proposed system and the EMD system are comparable because both of them use color features. The following three graphs show the comparison of their performance on the 10 categories of images.

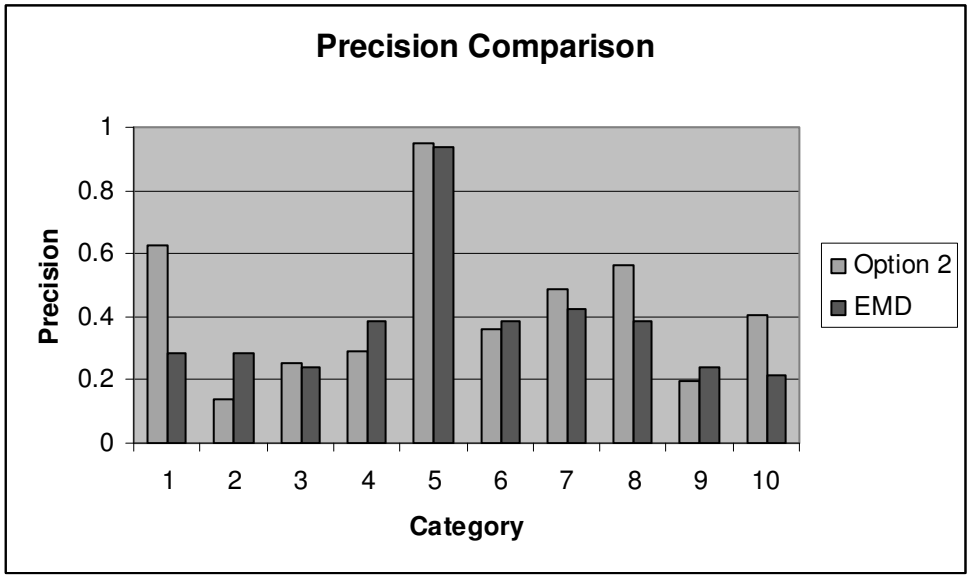


Figure 6.4 Precision Comparison between Option 2 and EMD

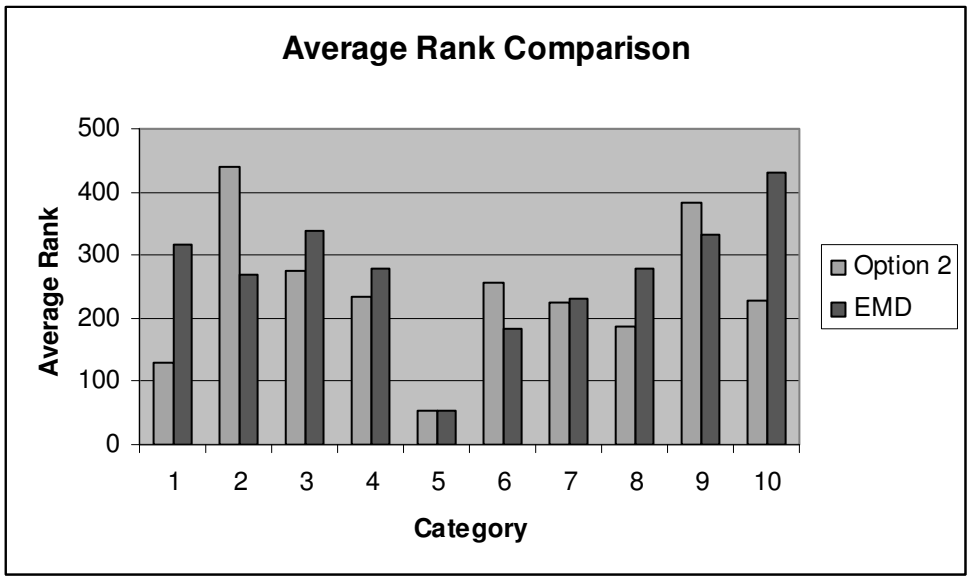


Figure 6.5 Average Rank Comparison between Option 2 and EMD

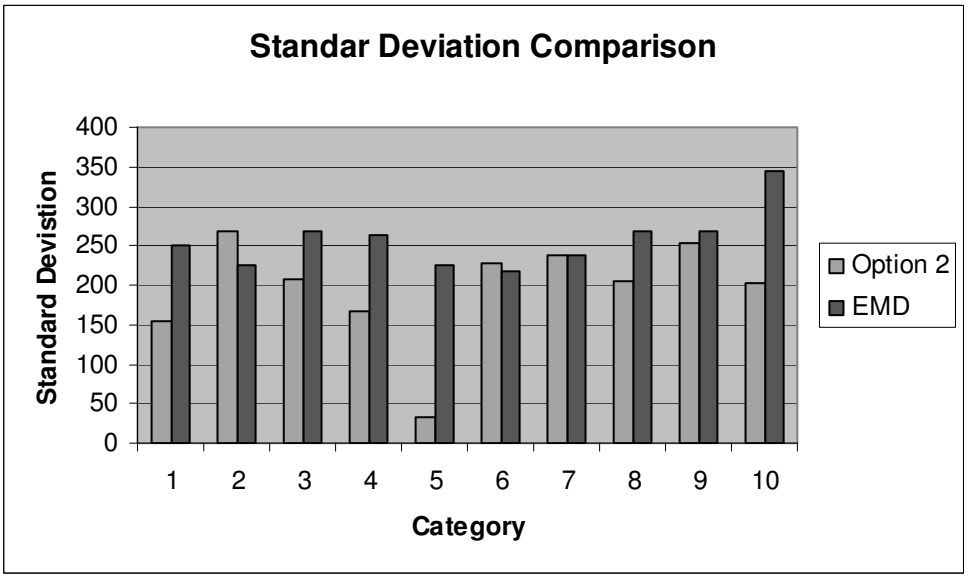


Figure 6.6 Standard deviation Comparison between Option 2 and EMD

System Name	Precision	Average Rank	Standard Deviation
Option 2	6 categories	7 categories	8 categories
EMD	4 categories	3 categories	2 categories

Table 6.2 Summary of Comparison of Option 2 and EMD

As summarized in Table 6.2 Option 2 is better than EMD.

6.3.3 Comparison with SIMPLIcity

The next comparison is between SIMPLIcity and the proposed system. The following three graphs show the comparison for the three statistics.

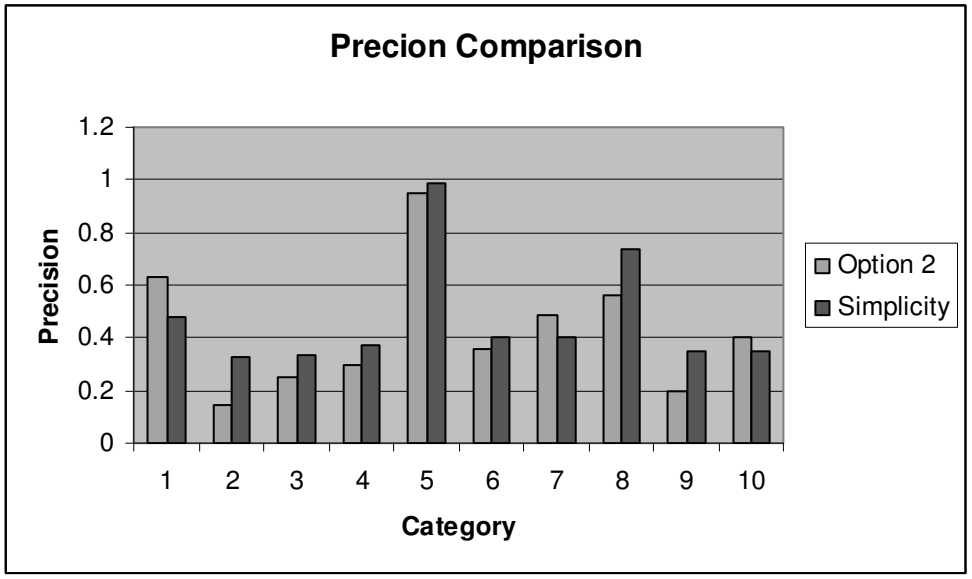


Figure 6.7 Precision Comparison between Option 2 and SIMPLIicty

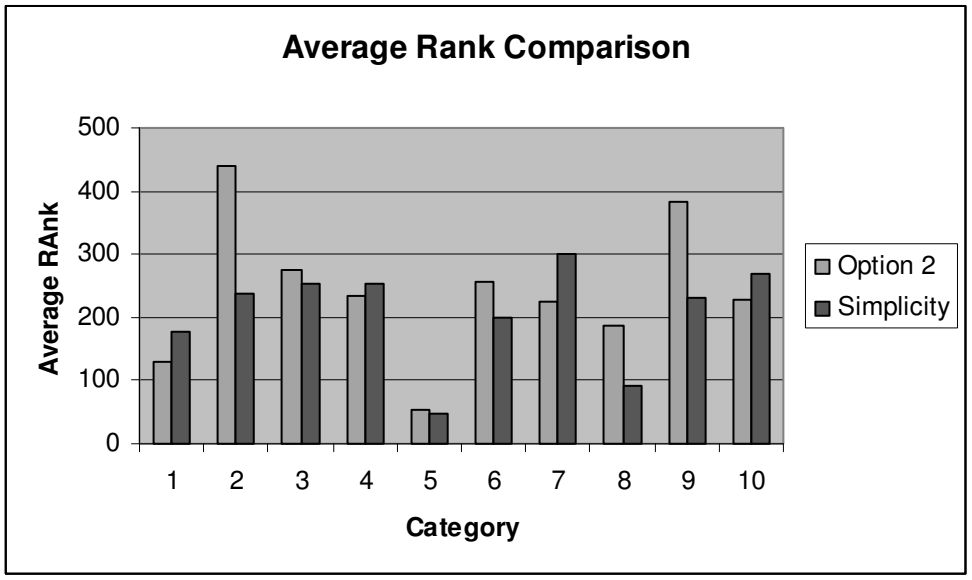


Figure 6.8 Average Rank Comparison between Option 2 and SIMPLIicty

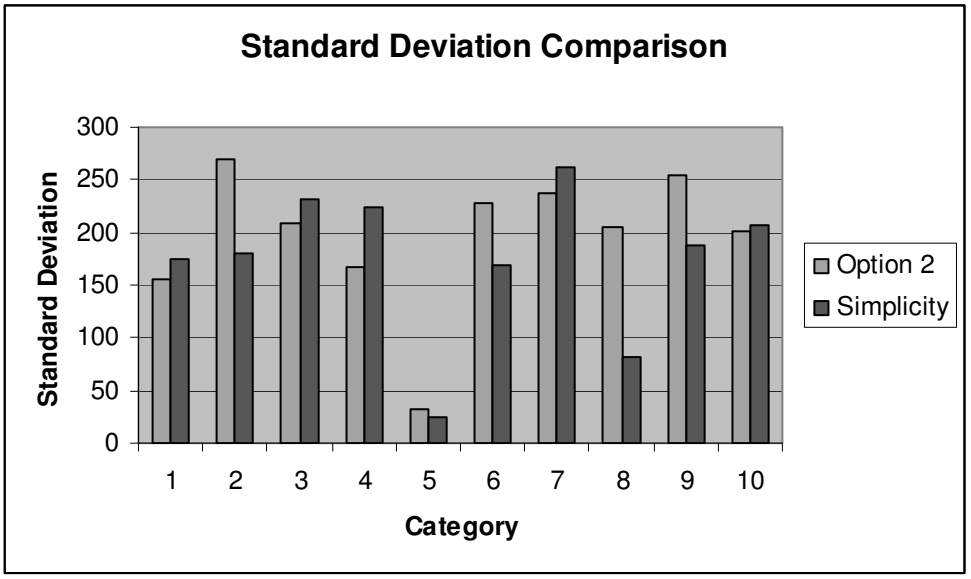


Figure 6.9 Standard Deviation Comparison between Option 2 and SIMPLIicty

System Name	Precision	Average Rank	Standard Deviation
Option 2	3 categories	4 categories	5 categories
SIMPLIcity	7 categories	6 categories	5 categories

Table 6.3 Summary of Comparison of Option 2 and SIMPLIcity

As shown in the Table 6.3, SIMPLIcity performs well in 7 of the 10 categories and the proposed system performs well in 3 of the categories. In the average rank, SIMPLIcity performs well in 6 of the 10 categories and the proposed system performs well in 4 the categories. In the standard deviation, both work well at 5 categories of the 10 categories.

One main reason for Option 2 to perform lower than SIMPLIcity is the feature used. It is difficult to represent all image types using a single feature. SIMPLIcity uses color, texture and shape features to represent images. The proposed system uses only color feature to represent images.

SIMPLIcity is an image retrieval system, which uses semantics classification methods, a wavelet-based approach for feature extraction, and integrated region matching based upon image segmentation. As in other region-based retrieval systems, an image is represented by a set of regions, roughly corresponding to objects, which are characterized

by color, texture, shape, and location.). The classification method relies on prior knowledge rather than training, and hence is not set up for extension. [1] The performance of SIMPLIcity is improved by the usage of classification and more number of features. Similarly, it is possible to improve the performance of the new system by using additional features and indexing.

6.3.4 Conclusion on Accuracy Tests

As the proposed system uses color feature alone, its performance is promising. The performance of the proposed system is low in category 2 and 3. Category 2 is a collection of images that show people at Beaches. A sample image from the category is shown in figure 6.7. As can be seen in the image, beaches contain sands which are highly textured.

It is difficult to segment highly textured images using color alone. The segmentation result for the image in figure 6.10 is shown in figure 6.11. Of the 3469 total regions, only 70 are effective regions. Small part of the image is stored in the database. This is because the textured sand areas can not be merged into regions of larger size by using the color feature alone. The performance for such kinds of pictures can be improved by using texture feature. The same explanation holds good for category 3.

Figure 6.10 Sample Image from beach category

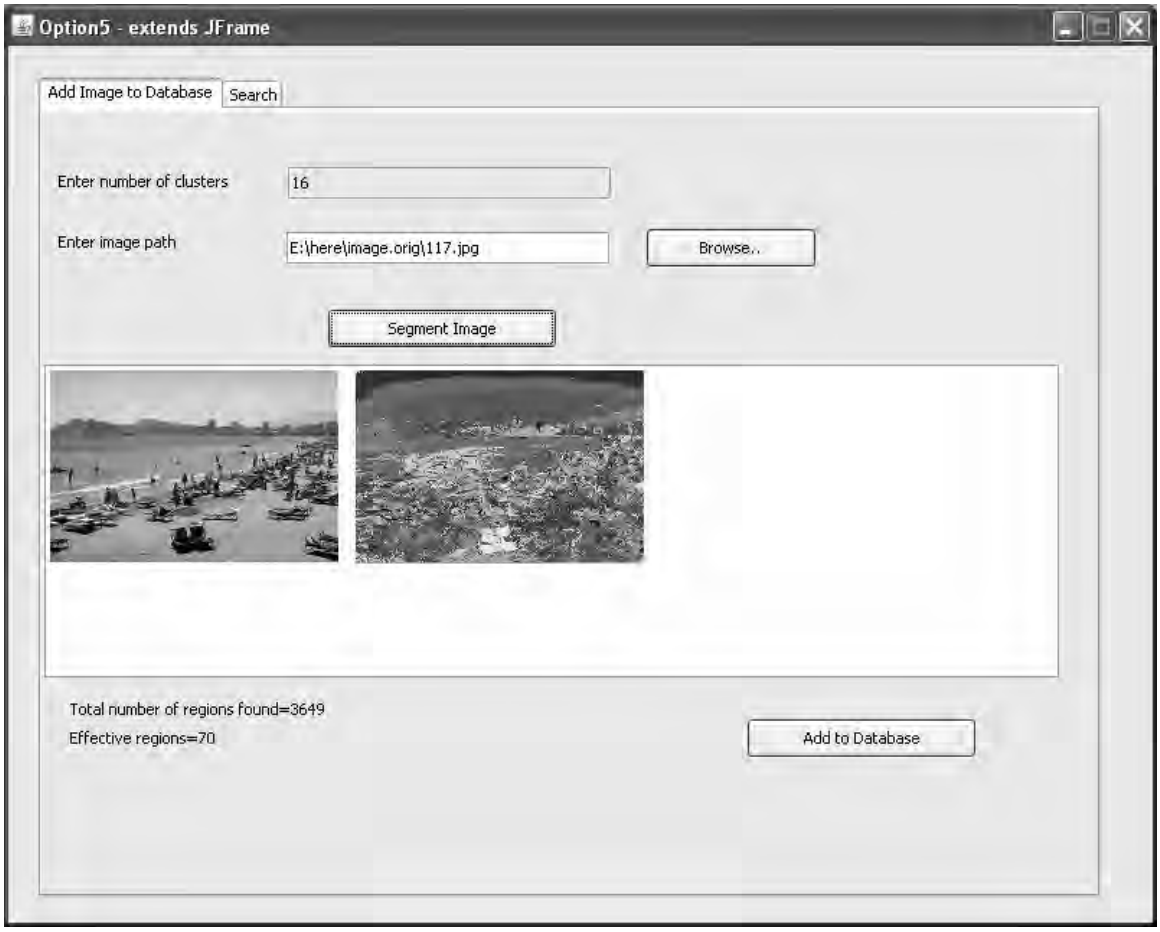


Figure 6.11 Segmentation of Textured Images

6.4 Speed Test

Next the proposed system is tested with 10,000 images that are used for testing the WBIIS system. It takes 40 seconds on average to search the 10,000 images on a PC with RAM of 1GB and processing speed of 3.39 GHz with Pentium 4 HT processor. The speed test was done using Windows XP operating system. Since the databases used for measuring the speed of the other systems are different, it is difficult to compare the speed of the proposed system with SIMPLIcity and EMD.

The speed of the system is not satisfactory. The speed of the proposed system can be improved by using classification (indexing) because classification reduces the number of images to be compared with the sample image. In addition some optimization work is needed on the program itself to reduce the time taken for the searching. For example, structures like HashMap that are provided by Java can be replaced with other convenient structures that are faster. The use of a simple DBMS like Microsoft Access may also have a negative impact on the speed. For example in reference [36], the performance of MS Access is compared with that of MySQL and it is said that MySQL is faster than MS Access. The programming language, Java has also its own impact on the speed.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis work, a CBIR system that is based on unreliable segmentation is developed. The result of the unreliable segmentation is to divide the image into regions far from semantic objects. This helps the system to overcome the drawbacks of the segmentation which is a difficult problem to solve analytically. It is very difficult to have a segmentation technique that works equally well for all image types. Hence expecting segmentation algorithm to divide images into few regions that are approximately equivalent to semantic objects may severely affect the performance of the system over a large type of images.

As the number of regions are too many it is mandatory to represent each regions in a feature database with small dimensionality. For this reason the regions are represented by their average color in the RGB space. The average color is calculated for the three channels separately as is shown in equation 4.1. Along with the average colors, the number of pixels of each region is also part of the feature database.

The similarity measurement is done based on the similarity of regions. Two regions are said to be similar if the Euclidean distance between their average color is less or equal to 20 and the ratio of number of pixels is less or equal to 1.5. Then image similarity will be based on either of the two options: on number of similar region counts or on sum of pixels in the similar regions.

The performance of the system is measured with three parameters: precision, average rank and standard deviation of the ranks. Based on these parameters, the system is compared with two systems: EMD and SIMPLIcity. The performance of the system was better than EMD but lesser than SIMPLIcity. More emphasis should be given to that of EMD because both systems are based on color feature alone.

7.2 Future Work

The following things can be added to the system to improve its performance.

- The system can be improved by using additional features like texture
- Instead of leaving out regions whose size is less than 0.1% of the total image size, it will be better if they are merged to neighboring regions whose size is greater than 0.1%
- Using color spaces like LUV and HSV which are more widely used in CBIR systems, may improve its performance.
- The speed can be improved by using dimension reduction technique and indexing schemes like R*-tree [37], linear quad-trees [38], K-d-B tree [39] and grid files [40].
- Using other DBMS software like Microsoft SQL server and Oracle may improve the speed.
- K-means cluster centers can be made adaptive so that the number of centers varies based on the image content during segmentation.

Appendix A

GUI for Segmentation

The interface for the segmentation part of the program is shown in figure A.1. The browse button is used to locate the path of the image file. After locating the path of the image file, the user can click the segment image button to segment the image. Both the original and the segmented output are displayed in the panel below the segment image button as shown in figure 3.7.

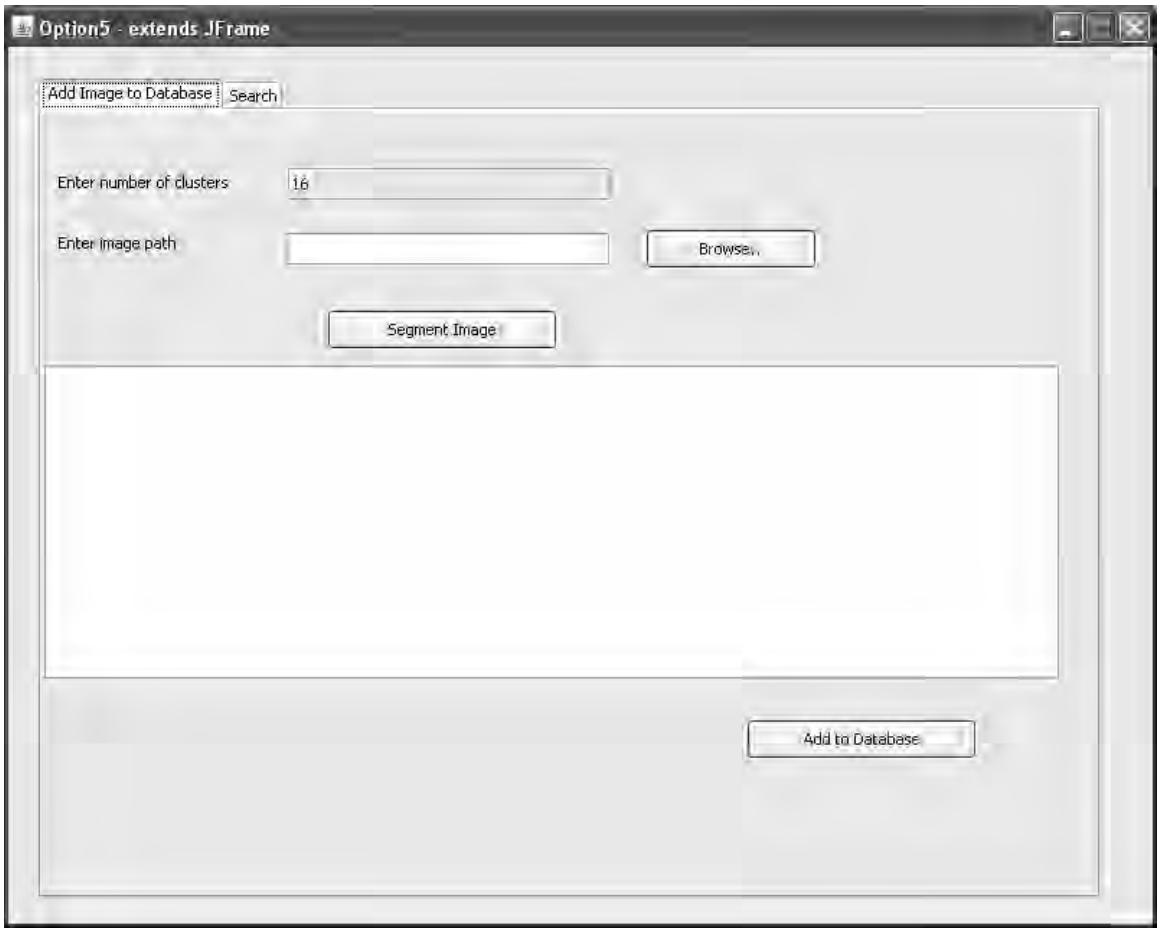


Figure A.1 GUI for Segmentation

Appendix B

GUI for Feature Extraction

Feature extraction is done after the image is segmented into regions. The user clicks the Add to Image Database button to add the color moments and number of pixels of all the effective regions to the database. A message box appears to indicate the successful addition of the image to the database. This is shown in figure B.1.

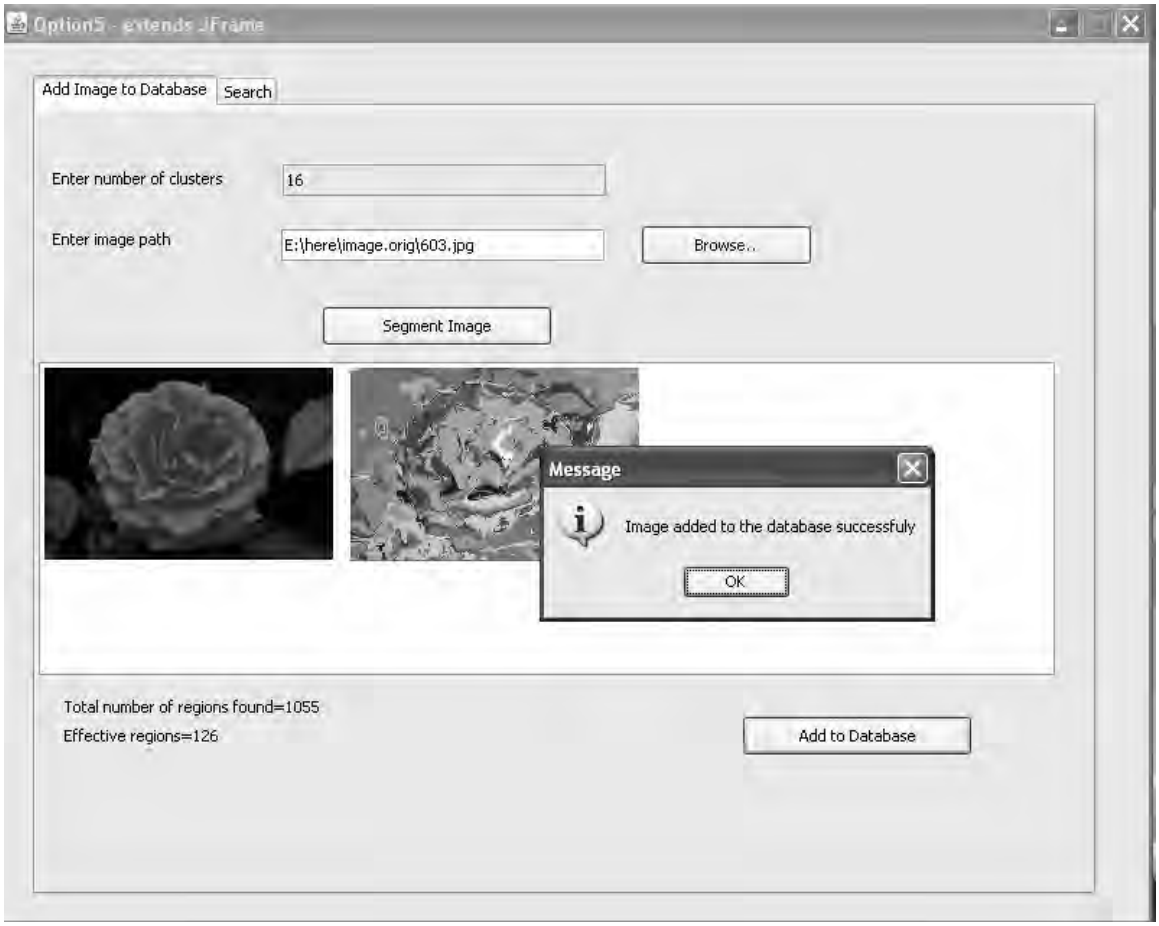


Figure B.1. GUI for feature extraction

Currently Microsoft Access Database is used to store the feature database. In the feature database, an image is represented by a table that contains the average color and the number of pixels of the effective regions of the image. Figure B.2 shows the partial list of the tables in the Microsoft Access database. For example table 604 stores the feature of image 604.

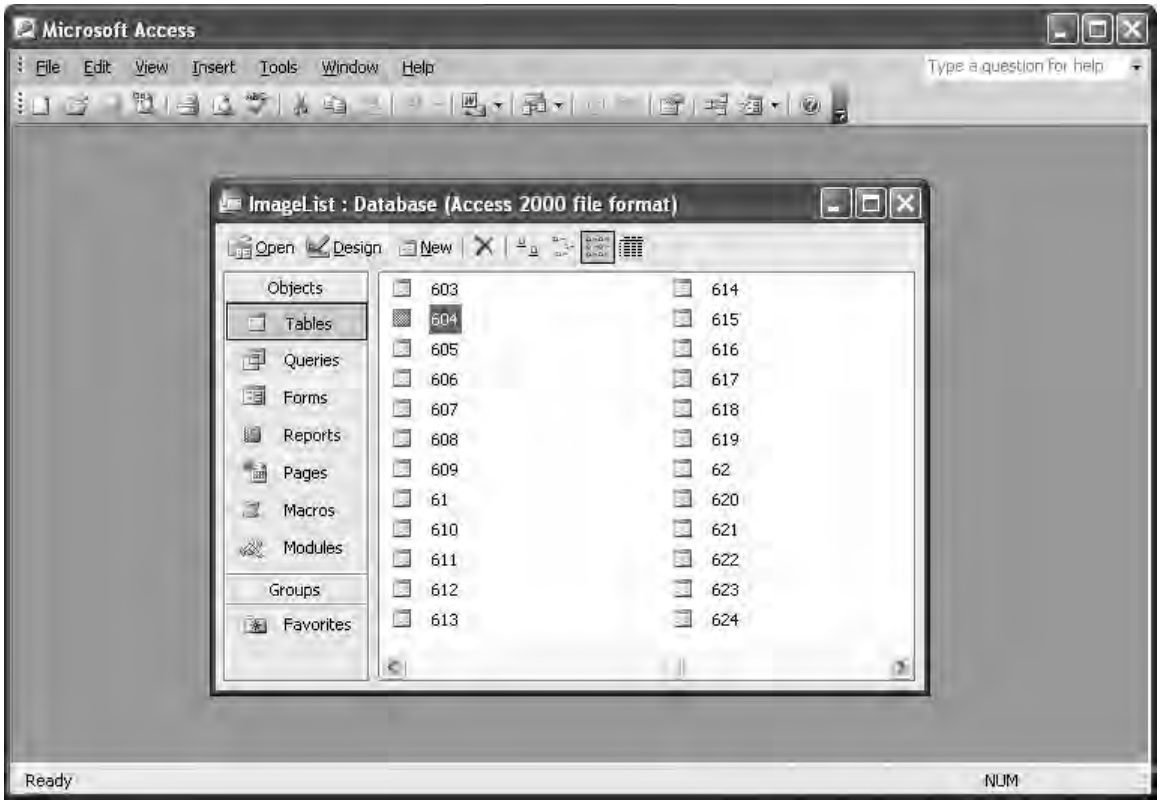


Figure B.2. Access Database

Each table has four fields. Three of them contain the average of the red, green and blue color channels of the region. The fourth field is the number of pixels of the region. Table B.1 displays the table for the image shown in figure B.1, the rose image. Each of the 126 effective regions, are represented by average values of red, green and blue colors and the number of pixels of the region.

Region No	**red**	**green**	**blue**	**noOfPixels**
1	2	2	2	96
2	2	2	1	41
3	7	7	7	67
4	3	2	1	73
5	7	7	7	553
6	201	91	89	30
7	3	3	1	195
8	208	23	13	54
9	203	19	15	90
10	13	4	5	30
11	201	91	88	41

12	6	6	6	93
13	33	50	37	35
14	14	2	4	38
15	17	23	23	37
16	218	40	30	214
17	228	64	56	215
18	203	19	13	34
19	234	61	55	37
20	50	67	56	80
21	197	86	83	39
22	217	42	33	163
23	10	8	6	56
24	51	66	53	95
25	64	81	72	703
26	4	4	4	904
27	229	66	59	95
28	225	61	54	47
29	14	16	15	380
30	204	18	15	28
31	218	41	26	64
32	20	17	11	29
33	12	4	6	45
34	230	61	45	27
35	162	3	5	160
36	190	4	3	250
37	9	9	8	496
38	202	14	13	56
39	163	1	3	29
40	227	61	54	138
41	233	61	55	84
42	207	46	38	57
43	229	62	54	47
44	218	38	27	31
45	186	4	2	90
46	205	18	12	429
47	129	2	3	30
48	217	43	38	73
49	227	47	76	32
50	202	25	12	31
51	14	17	14	111
52	154	5	6	77
53	218	39	36	102
54	70	86	74	228
55	3	3	1	233
56	196	26	20	82
57	1	3	2	40
58	44	61	54	569
59	215	40	34	30

60	50	68	52	237
61	205	18	12	98
62	188	25	27	97
63	2	2	1	43
64	210	41	30	35
65	182	3	4	1690
66	201	14	20	39
67	202	17	13	31
68	200	39	39	30
69	3	3	1	72
70	220	44	31	356
71	163	1	5	49
72	4	4	4	2636
73	231	65	64	77
74	158	1	4	205
75	3	3	3	1603
76	209	43	38	50
77	180	5	5	613
78	36	43	34	58
79	206	21	16	83
80	225	59	58	40
81	230	69	61	92
82	2	3	1	39
83	181	4	5	131
84	205	19	12	516
85	3	3	1	43
86	190	25	23	34
87	201	21	17	304
88	21	29	23	278
89	227	61	52	60
90	118	4	11	239
91	217	44	32	446
92	227	56	45	33
93	20	33	26	27
94	13	7	6	43
95	208	40	44	127
96	153	12	21	53
97	153	12	19	264
98	35	41	32	59
99	13	19	16	185
100	14	17	12	51
101	31	43	41	454
102	4	4	4	31
103	8	10	7	120
104	13	21	15	78
105	150	7	15	194
106	12	20	15	225
107	2	3	1	29

108	57	7	11	29
109	34	50	41	168
110	3	4	4	90
111	21	35	22	36
112	4	4	4	121
113	1	1	0	241
114	5	4	4	62
115	4	4	5	32
116	31	48	35	52
117	8	8	9	29
118	10	8	6	72
119	8	12	7	193
120	12	20	9	44
121	13	17	17	83
122	20	28	26	244
123	122	4	11	452
124	32	44	39	147
125	20	31	24	432
126	61	7	10	86

Table B.1. Sample Feature table

Appendix C

GUI for Image Retrieval

The interface for the image search is shown in figure C.1 below.

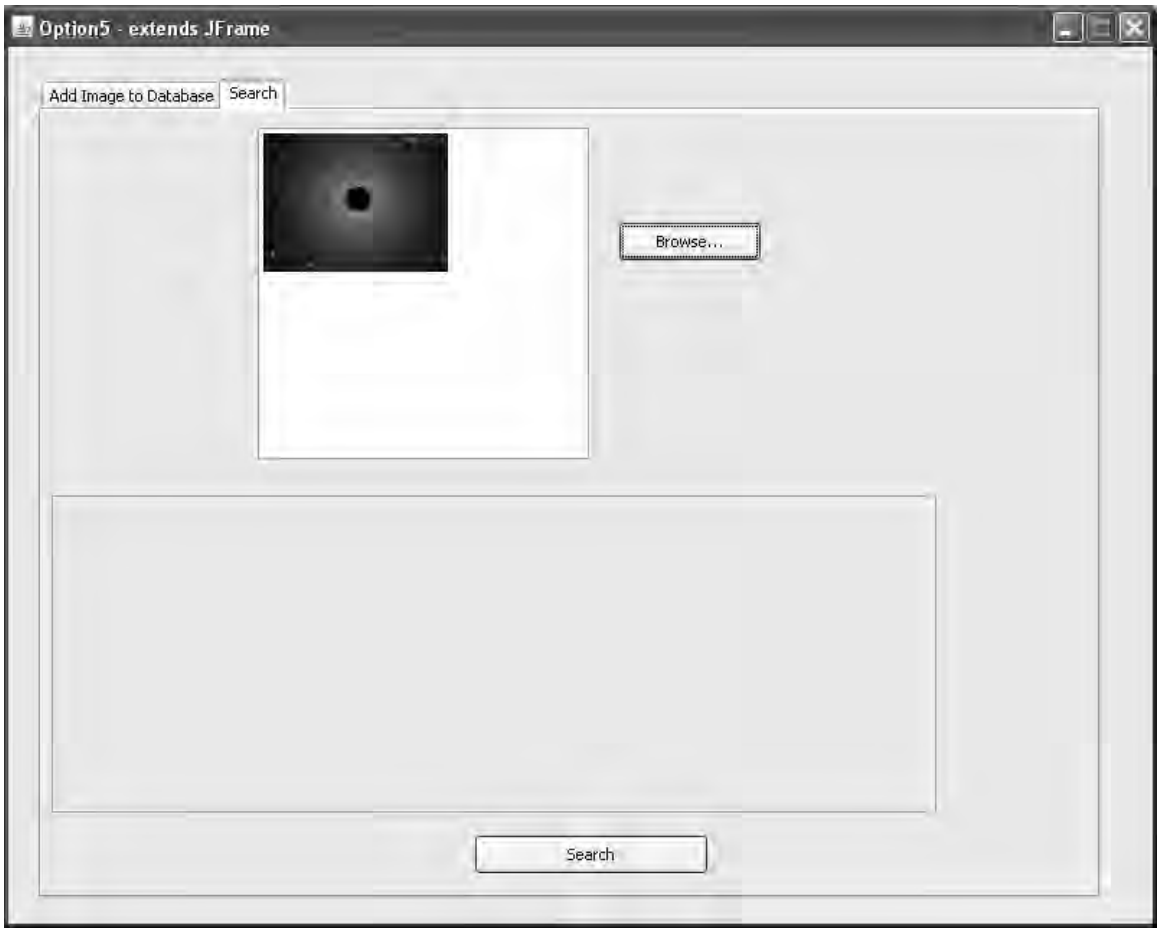


Figure C.1 Image search interface

The browse button is used to select the sample image. Once the sample image is selected, the search button is used to do the retrieval. The system returns the top 100 images in the pane below the sample image. This is shown in figure C.2. The user can scroll to see the hidden images.

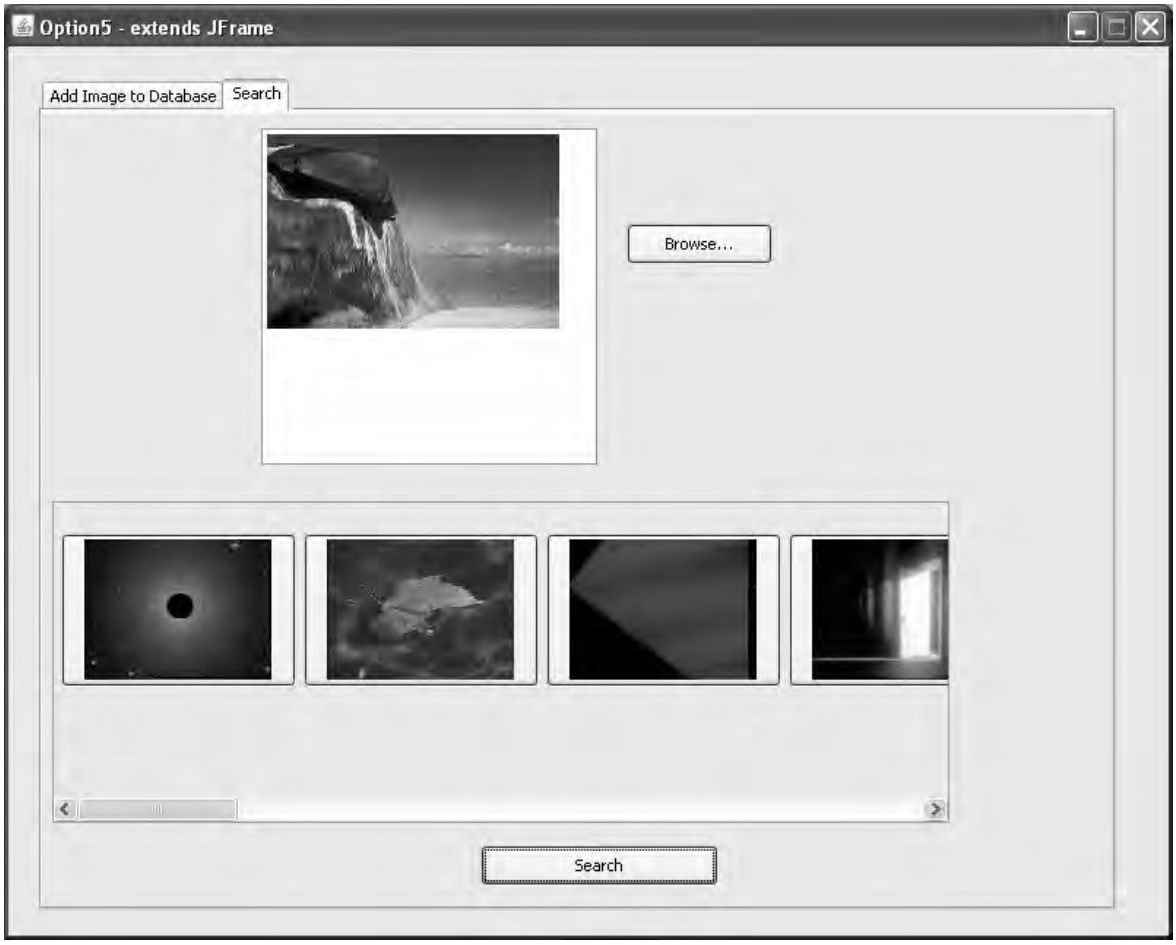


Figure C.2 Retrieval Output

Appendix D

Source Code

D.1 Segmentation

```
/**
 * @author Shewatatek Lemma
 */
package CBIR;

import java.awt.image.BufferedImage;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.IOException;
import java.awt.Color;
import java.awt.Point;
import java.sql.*;
import java.util.HashMap;
import java.util.Iterator;

public class SegmentImageNew
{
   BufferedImage im ;
   Pixel[] pixels;
   int width,height,clusterNum,band;
   double[][] clusters;
   double[][] distance;
   int[] count;
   double[][] sum;
   int rAll;
   HashMap<Integer,Region> region;
   Statement s;
   int eRegion = 0;
   int totalNumberOfPixels;
   public SegmentImageNew(BufferedImage im,int clusterNum,int band)
   {
         this.im = im;
         this.clusterNum = clusterNum;
         this.band = band;
         extractColor();
   }
   void extractColor()
   {
         width = im.getWidth();
```

```
        height = im.getHeight();
        totalNumberOfPixels = width * height;
        pixels = new Pixel[totalNumberOfPixels];
        int t,pixel;
        for(int j=0;j<height;j++)
        {
                for(int i=0;i<width;i++)
                {
                t = j * width + i;
           pixels[t] = new Pixel();
           pixels[t].bandValues = new int[3];
                pixel = im.getRGB(i,j);
                        pixels[t].bandValues[0] =  (pixel >> 16) & 0xff; //Red component
                        pixels[t].bandValues[1] =  (pixel >>  8) & 0xff;//Green component
                        pixels[t].bandValues[2] =  (pixel      ) & 0xff;//Blue
                        pixels[t].x = i;
                        pixels[t].y = j;
           pixels[t].regionNum = -1;
           pixels[t].isBorder = false;
                }
        }
}
public void cluster()
{
        initializeClusterCenters();
        count = new int [clusterNum];
        boolean noChange = true;
        sum = new double[clusterNum][band];
        int maxIteration = 30;
        for(int k=0;k<maxIteration;k++)
        {
                distanceCalc();
                for(int i=0;i<totalNumberOfPixels;i++)
                {
                        int x = min(distance[i]);
                        if(x!=pixels[i].group)
                        {
                                pixels[i].group = x;
                                noChange = false;
                        }
                }
                noChange = true;
                calcNewClusters();
        }
}
        void initializeClusterCenters()
```

```
        {
        clusters = new double[clusterNum][band];
        int k = 0;
        int g = (totalNumberOfPixels)/clusterNum;
        for(int i=0;i<clusterNum;i++)
        {
                clusters[i][0] = pixels[k].bandValues[0];
                clusters[i][1] = pixels[k].bandValues[1];
                clusters[i][2] = pixels[k].bandValues[2];
                k+=g;
        }
        }
    void distanceCalc()
    {
        double sum;
        distance = new double[totalNumberOfPixels][clusterNum];
        for(int i=0; i<totalNumberOfPixels;i++)
        {
                for(int k=0;k<clusterNum;k++)
                {
                        sum = 0;
                        for(int j=0;j<band;j++)
                        {
                                sum = sum + (pixels[i].bandValues[j]-
clusters[k][j])*(pixels[i].bandValues[j]-clusters[k][j]);
                        }
                        distance[i][k] = Math.sqrt(sum);
                }
        }
    }

    int min(double[] data)
    {
        int minIndex = 0;
        for(int i=0;i<data.length;i++)
        {
                if(data[i]<data[minIndex])
                        minIndex = i;
        }
        return minIndex;
    }

    void calcNewClusters()
    {
       for(int i=0;i<clusterNum;i++)
       {
```

```
            count[i]=0;
            for(int j=0;j<band;j++)
            {
               sum[i][j]=0;
            }
         }
            for(int i=0;i<totalNumberOfPixels;i++)
            {
                     sum[pixels[i].group][0] = sum[pixels[i].group][0] +
pixels[i].bandValues[0];
                     sum[pixels[i].group][1] = sum[pixels[i].group][1] +
pixels[i].bandValues[1];
                     sum[pixels[i].group][2] = sum[pixels[i].group][2] +
pixels[i].bandValues[2];
                     count[pixels[i].group]++;
            }
            for(int i=0; i<clusterNum;i++)
            {
                     if(count[i]!=0)
                     {
                              clusters[i][0] = sum[i][0]/count[i];
                              clusters[i][1] = sum[i][1]/count[i];
                              clusters[i][2] = sum[i][2]/count[i];
                     }
                     else
                     {

                     }
            }
      }
      void identifyRegions()
      {
         Point[] v = new Point[4];
         for(int i=0;i<v.length;i++)
            v[i] = new Point();
         region = new HashMap();
         int index1,index2,r;
         rAll = 0;
         for(int j=0;j<height;j++)
         {
                  for(int i=0;i<width;i++)
                  {
               index1 = j * width + i;
               r = pixels[index1].regionNum;
               if(r==-1)
               {
```

```
            r = rAll;
            pixels[index1].regionNum = r;
            pixels[index1].isBorder = true;
            Region t = new Region();
            t.number = r;
            t.pixels = new StringBuffer();
            t.borderPixels = new StringBuffer();
            t.pixels.append(index1 + ",");
            t.borderPixels.append(index1 + ",");
            t.noOfPixels = 1;
            region.put(r,t);
            rAll++;
          }
          findNeighbor(i,j,v);
          for(int k=0;k<v.length;k++)
          {
            index2 = v[k].y * width + v[k].x;
            if(v[k].x == i && v[k].y == j)
            {

            }
            else if(pixels[index1].group == pixels[index2].group )
            {
              if(pixels[index2].regionNum==-1)
              {
                pixels[index2].regionNum = r;
                Region t = region.get(r);
                t.pixels.append(index2 + ",");
                        t.noOfPixels++;
              }
              else
              {
                    if(pixels[index1].regionNum != pixels[index2].regionNum)

      merge(pixels[index1].regionNum,pixels[index2].regionNum);
              }
            }
          }
            }
    }
            //writeToFile("beforeMerge.txt");
}
      void findNeighbor(int i, int j,Point[] c)
      {
            if(i+1<width)
            {
```

```
                c[0].x = i+1;
                c[0].y = j;
        }
        else
        {
                c[0].x = i;
                c[0].y = j;
        }
        if( j+1<height)
        {
                c[1].x = i;
                c[1].y = j+1;
        }
        else
        {
                c[1].x = i;
                c[1].y = j;
        }
        if( j+1<height && i-1>-1)
        {
                c[2].x = i-1;
                c[2].y = j+1;
        }
        else
        {
                c[2].x = i;
                c[2].y = j;
        }
        if( j+1<height && i+1<width)
        {
                c[3].x = i+1;
                c[3].y = j+1;
        }
        else
        {
                c[3].x = i;
                c[3].y = j;
        }
}
void merge(int r1, int r2)
{
        Region t1 = region.get(r1);
        Region t2 = region.get(r2);
        t1.pixels.append(t2.pixels);
        t1.noOfPixels += t2.noOfPixels;
        region.remove(r2);
```

```
                adjustRegionNum(t2.pixels,r1);
        }
        void storeInDB()
        {
        try
        {
                s = DatabaseConnection.con.createStatement();
                s.executeUpdate("Delete From data");
                s.close();
                s = DatabaseConnection.con.createStatement();
                eRegion = 0;
                Iterator<Integer> it = region.keySet().iterator();
                int index2;
                while(it.hasNext())
                {
                        index2 = it.next();
                        if(region.get(index2).noOfPixels>0.001*totalNumberOfPixels)
//Region is greator that 0.1% of the total image area
                        {
                                s.execute("insert into data  values('" +
region.get(index2).pixels.toString() + "');");
                                eRegion++;
                        }
                }
                s.close();
        }
                catch(SQLException e)
        {
                        System.out.println(e.getMessage());
        }
        }
        void adjustRegionNum(StringBuffer st, int newRegionNum)
        {
                String[] pix = st.toString().split(",");
                int x;
                for(int i=0;i<pix.length;i++)
                {
                        x = Integer.parseInt(pix[i]);
                        pixels[x].regionNum = newRegionNum;
                }
        }
        int[] draw()
        {
                //rAll = region.size();
        Color[] color = new Color[rAll];
        for(int i=0;i<rAll;i++)
```

```
        {
                color[i] = new Color((int)(Math.random() *
255),(int)(Math.random()*255) ,(int)(Math.random() * 255));
        }
        int[] segPixels = new int[totalNumberOfPixels];
        for(int i=0; i<totalNumberOfPixels;i++)
        {
                segPixels[i] = color[pixels[i].regionNum].getRGB();
        }

    return segPixels;

        }
}
```

D.2 Feature Extraction

```
/**
 * @author Shewatatek Lemma
 */
package CBIR;

import java.io.*;
import javax.swing.*;
import java.sql.*;
public class ColorMoments
{
        Pixel[] pixels;
        int width;
        int height;
        double[][] moments;
        Statement s,s1;
        String fileName;
        public ColorMoments(int eRegion,Pixel[] pixels, int width,int height,String
fileName) throws SQLException
        {
                try
                {
        String st ="create table ["+fileName+"] (red Integer, green Integer, blue
Integer,noOfPixels Integer);" ;
              s = DatabaseConnection.con.createStatement();
        s.execute(st);
        System.out.println(st);
                }
                catch(SQLException e)
                {
        System.out.println(e.getMessage());
        System.exit(0);
                }

      this.width = width;
      this.height = height;
      this.pixels = pixels;
      this.fileName = fileName;

      moments = new double[eRegion][3];
      String x;
      boolean moreRow=true;
      int rCount = 0;
      String[] x1;
```

```
        s1 = DatabaseConnection.con.createStatement (
ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
        s1.execute("select * from data");
        ResultSet rs = s1.getResultSet();
        rs.first();
        while(moreRow)
        {
           x = rs.getString("Pixel");
           x1 = x.split(",");
           calcMoments(x1,rCount);
           rCount++;
           moreRow = rs.next();
        }
        s1.close();
        s.close();
           }
           void calcMoments(String[] x,int rCount) //throws IOException
           {
                    double sumRed = 0;
                    double sumGreen = 0;
                    double sumBlue = 0;
        int i;
                    for(int k=0;k<x.length;k++)
                    {
                             i = Integer.parseInt(x[k]);
                             sumRed = sumRed + pixels[i].bandValues[0];
           sumGreen = sumGreen + pixels[i].bandValues[1];
                             sumBlue = sumBlue + pixels[i].bandValues[2];
                    }
                    int h = x.length;
                    moments[rCount][0] = sumRed/(h);
                    moments[rCount][1] = sumGreen/(h);
                    moments[rCount][2] = sumBlue/(h);
        try
        {
                  s.execute("insert into [" + fileName + "] values (" + moments[rCount][0] +
"," + moments[rCount][1]+ "," + moments[rCount][2] +"," + h + ");");
        }
        catch(SQLException e)
        {
           System.out.println(e.getMessage());
        }
           }
}
```

D.3 Similarity Measure – Region Count

```
/**
 * @author Shewatatek Lemma
 */
package CBIR;

import java.sql.*;
import java.io.*;
import javax.swing.*;
import java.util.*;
public class Distance
{

   String[] CMFiles;
   double[][] sample;
   ImagePath[] ip;
   //Region[] a;
   double[] ss1;
   public Distance(String sampleName,int eRegion, HashMap<Integer,Region> region)
   {
                Iterator<Integer> it = region.keySet().iterator();
                //a = new Region[region.size()];
                int index2;
                int z=0;
        while(it.hasNext())
        {
                index2 = it.next();
                        //a[z] = region.get(index2);
                        z++;
        }
      try
      {
        //this.a = a;
        Statement s =
DatabaseConnection.con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,Re
sultSet.CONCUR_UPDATABLE);
        s.execute("Select * From List");
        ResultSet rs = s.getResultSet();
        rs.last();
        int x = rs.getRow();
        CMFiles = new String[x];
        ip = new ImagePath [x];
        rs.first();
        boolean moreRow=true;
        int i=0;
        while(moreRow)
```

```
        {
           CMFiles[i] = rs.getString("CMFileName");
           ip[i] = new ImagePath();
           ip[i].path = rs.getString("Path");
           ip[i].count = new double[Integer.parseInt(rs.getString("NoOfRegions"))];
           i++;
           moreRow = rs.next();
        }
        s.execute("select * from ["+sampleName+"] ;");
        rs = s.getResultSet();
        boolean hasMore = true;
        sample = new double[eRegion][4];
        ss1 = new double[4];
        rs.first();
        try
        {
           i=0;
           while(hasMore)
           {
              sample[i][0] = rs.getDouble("red");
              sample[i][1] = rs.getDouble("green");
              sample[i][2] = rs.getDouble("blue");
              sample[i][3] = rs.getDouble("noOfPixels");
              hasMore = rs.next();
              i++;
           }

        }
        catch(SQLException e)
        {

        }
     }
     catch (SQLException e)
     {
        System.out.println("Error: " + e);
     }

}
ImagePath[] findImages()
{
     int d,t1,t2;
     double p,m;
     Hashtable t = new Hashtable();
     Hashtable h1 = new Hashtable();
     try
```

```
      {
        Statement st =
DatabaseConnection.con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,Re
sultSet.CONCUR_UPDATABLE);
        ResultSet rs;
                //Read a region and compare it with regions of the sample image
        for(int k=0;k<CMFiles.length;k++)
        {
                st.execute("select * from ["+ CMFiles[k]+"] ;");
                rs = st.getResultSet();
                t.clear();
                h1.clear();
                d=0;
                ip[k].sum=0;
                ip[k].similarPixels=0;
                boolean hasMore = true;
                rs.first();
                while(hasMore)
                {
                        ss1[0] = rs.getDouble("red");
                        ss1[1] = rs.getDouble("green");
                        ss1[2] = rs.getDouble("blue");
                        ss1[3] = rs.getDouble("noOfPixels");
                        d ++;

//System.out.println(ss1[0]+","+ss1[1]+","+ss1[2]+","+ss1[3]+"\t"+sample[0][0]+","+sa
mple[0][1]+","+sample[0][2]+","+sample[0][3]);
                        for(int i=0;i<sample.length;i++)
                        {
                                p = compare(ss1,sample[i]);
                                if(p<=20 && comparable(ss1[3],sample[i][3]))// &&
!t.containsKey(i))
                                {
                                        ip[k].sum++;
                                        ip[k].similarPixels += sample[i][3];
                                        t.put(i,p);
                                        h1.put(d,i);
                                        //break;
                                }
                        }
                   hasMore = rs.next();
                    }
                    ip[k].remainder = d-h1.size();
                rs.close();
              }
        ImagePath iptemp;
```

```
                for(int i=0;i<CMFiles.length-1;i++)
                {
                   for(int j=i+1;j<CMFiles.length;j++)
                   {
                      if(ip[i].sum<ip[j].sum)
                      {
                         iptemp = ip[i];
                         ip[i] = ip[j];
                         ip[j] = iptemp;
                      }
                   }
                }
        }
        catch(SQLException e)
        {
           JOptionPane.showMessageDialog(null,e.getMessage());
        }
        return ip;
    }
    boolean comparable(double s, double t)
    {
          if(s>t)
                return (double)s/t<=1.5;
          else
                return (double)t/s<=1.5;
    }
    double compare(double[] s1, double[] s2)
    {
       return Math.sqrt((s1[0]-s2[0])*(s1[0]-s2[0])+(s1[1]-s2[1])*(s1[1]-s2[1])+(s1[2]-
s2[2])*(s1[2]-s2[2]));

    }
}
```

D.4 Similarity Measure – Pixel Sum

```
/**
 * @author Shewatatek Lemma
 */
package CBIR;

import java.sql.*;
import java.io.*;
import javax.swing.*;
import java.util.*;
public class Distance
{

   String[] CMFiles;
   double[][] sample;
   ImagePath[] ip;
   //Region[] a;
   double[] ss1;
   public Distance(String sampleName,int eRegion, HashMap<Integer,Region> region)
   {
                Iterator<Integer> it = region.keySet().iterator();
                //a = new Region[region.size()];
                int index2;
                int z=0;
        while(it.hasNext())
        {
                index2 = it.next();
                        //a[z] = region.get(index2);
                        z++;
        }
      try
      {
        //this.a = a;
        Statement s =
DatabaseConnection.con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,Re
sultSet.CONCUR_UPDATABLE);
        s.execute("Select * From List");
        ResultSet rs = s.getResultSet();
        rs.last();
        int x = rs.getRow();
        CMFiles = new String[x];
        ip = new ImagePath [x];
        rs.first();
        boolean moreRow=true;
        int i=0;
        while(moreRow)
```

```
        {
           CMFiles[i] = rs.getString("CMFileName");
           ip[i] = new ImagePath();
           ip[i].path = rs.getString("Path");
           ip[i].count = new double[Integer.parseInt(rs.getString("NoOfRegions"))];
           i++;
           moreRow = rs.next();
        }
        s.execute("select * from ["+sampleName+"] ;");
        rs = s.getResultSet();
        boolean hasMore = true;
        sample = new double[eRegion][4];
        ss1 = new double[4];
        rs.first();
        try
        {
           i=0;
           while(hasMore)
           {
              sample[i][0] = rs.getDouble("red");
              sample[i][1] = rs.getDouble("green");
              sample[i][2] = rs.getDouble("blue");
              sample[i][3] = rs.getDouble("noOfPixels");
              hasMore = rs.next();
              i++;
           }

        }
        catch(SQLException e)
        {

        }
     }
     catch (SQLException e)
     {
        System.out.println("Error: " + e);
     }

}
ImagePath[] findImages()
{
     int d,t1,t2;
     double p,m;
     Hashtable t = new Hashtable();
     Hashtable h1 = new Hashtable();
     try
```

```
      {
        Statement st =
DatabaseConnection.con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,Re
sultSet.CONCUR_UPDATABLE);
        ResultSet rs;
                //Read a region and compare it with regions of the sample image
        for(int k=0;k<CMFiles.length;k++)
        {
                st.execute("select * from ["+ CMFiles[k]+"] ;");
                rs = st.getResultSet();
                t.clear();
                h1.clear();
                d=0;
                ip[k].sum=0;
                ip[k].similarPixels=0;
                boolean hasMore = true;
                rs.first();
                while(hasMore)
                {
                        ss1[0] = rs.getDouble("red");
                        ss1[1] = rs.getDouble("green");
                        ss1[2] = rs.getDouble("blue");
                        ss1[3] = rs.getDouble("noOfPixels");
                        d ++;

//System.out.println(ss1[0]+","+ss1[1]+","+ss1[2]+","+ss1[3]+"\t"+sample[0][0]+","+sa
mple[0][1]+","+sample[0][2]+","+sample[0][3]);
                        for(int i=0;i<sample.length;i++)
                        {
                                p = compare(ss1,sample[i]);
                                if(p<=20 && comparable(ss1[3],sample[i][3]))// &&
!t.containsKey(i))
                                {
                                        ip[k].sum++;
                                        ip[k].similarPixels += sample[i][3];
                                        t.put(i,p);
                                        h1.put(d,i);
                                        //break;
                                }
                        }
                   hasMore = rs.next();
                    }
                    ip[k].remainder = d-h1.size();
                rs.close();
              }
        ImagePath iptemp;
```

```
                for(int i=0;i<CMFiles.length-1;i++)
                {
                   for(int j=i+1;j<CMFiles.length;j++)
                   {
                      if(ip[i]. similarPixels <ip[j]. similarPixels)
                      {
                         iptemp = ip[i];
                         ip[i] = ip[j];
                         ip[j] = iptemp;
                      }
                   }
                }
      }
      catch(SQLException e)
      {
         JOptionPane.showMessageDialog(null,e.getMessage());
      }
      return ip;
   }
   boolean comparable(double s, double t)
   {
         if(s>t)
                  return (double)s/t<=1.5;
         else
                  return (double)t/s<=1.5;
   }
   double compare(double[] s1, double[] s2)
   {
      return Math.sqrt((s1[0]-s2[0])*(s1[0]-s2[0])+(s1[1]-s2[1])*(s1[1]-s2[1])+(s1[2]-
s2[2])*(s1[2]-s2[2]));

   }
}
```

References

1. John Eakins, Margaret Graham, **"Content-based Image Retrieval"**, University of Northumbria at Newcastle**,** Technical Report, 1999.
2. Rafael C. Gonzalez, Richard E. Woods, **"Digital Image Processing"***,* PEARSON Printice Hall,2003
3. D. A. Forsyth, J. Ponce, **"Computer Vision: A modern Approach"**, Prentice-Hall, 2003.
4. Ritendra Datta, Dhiraj Joshi, Jia Li and James Z. Wang, **"Image Retrieval: Ideas, Influences, and Trends of the New Age,"** ACM Computing Surveys**,** vol. 40, 2007.
5. **Content-based Image retrieval**,http://en.wikipedia.org/wiki/Content-based_image_retrieval [Last viewed February 05, 2007]
6. Irena Valova, Boris Rachev, **"Retrieval by Color Features in Image Databases"**, Advances in Databases and Information Systems (Local proceedings) , Budapest Hungary, 2004
7. R. C. Veltkamp and M. Tanase. **Content-Based Image Retrieval Systems: A Survey**. Technical Report UU-CS-2000-34, Institute of Information and Computing Sciences, Utrecht University, 2002
8. Flickner, M., Sawhney, H., Niblack, W., Ashley, J., et al: **Query by image and video content: The QBIC system**. IEEE Computer, 28, Sept. 1995
9. Jie Luo, "**Content-Based Sub-Image Retrieval Using Relevance Feedback**", ACM International Workshop on Multimedia Databases**,** 2004: Arlington, VA, USA
10. C. Carson, M. Thomas, S. Belongie, J.M. Hellerstein, and J. Malik, **"Blobworld: A System for Region-Based Image Indexing and Retrieval"**, Proc. **Visual Information Systems**, pp. 509-516, June 1999.
11. Julien Fauqueur and Nozha Boujemaa, **"Region-Based Image Retrieval: Fast Coarse Segmentation and Fine Color Description"**, Journal of Visual Languages and Computing, Volume 15, 2004

12. James Z. Wang, Jia Li, and Gio Wiederhold, **Simplicity: Semantics-sensitive integrated matching for picture libraries**. IEEE Trans. Pattern Analysis and Machine Intelligence, 2001
13. S.K. Pal et al., **"A Review on Image Segmentation Techniques"**, Pattern Recognition, 29, 1277-1294, 1993
14. K.S. Fu and J.K. Mui, **"A Survey on Image Segmentation,"** Pattern Recognition 13, 3-16, 1981.
15. Pratt, William K., **"Digital Image Processing 3rd ed."**, John Wiley & Sons, Inc., 2001
16. H. D. Cheng, X. H. Jiang, Y. Sun, and Jing Li Wang, **Color image segmentation: Advances & prospects**. Elsevier Science, Pattern Recognition, 34(12):2259–2281, December 2001.
17. Jaume Vergés Llahí, ,**"Color Constancy and Image Segmentation Techniques for Applications to Mobile Robotics"**, **PHD Thesis, Universitat Politècnica de Catalunya**, April 2005
18. Ricardo da Silva Torres, Alexandre X. Falcão, **Content-Based Image Retrieval: Theory and Applications**. Revista de Informática Teórica e Aplicada, 13(2): 161-185,2006
19. Daniela Stan , **eID: A System For Exploration Of Image Databases**, PhD Thesis, Oakland University, Rochester, Michigan, 2002
20. Cotton S, 'Color, color spaces and the human visual system.' Technical report – CSR-95-03, Department of Computer Science, University of Birmingham , 1995
21. Wyszecki G. and Stiles W.S. (1982). *"Color science: concepts and methods, quantitative data and formulae"*, John Wiley & Sons, Inc., New York, NY, 2nd edition.
22. Symon D'O. Cotton, **Color, Color Spaces and the Human Visual System**, Technical Report, B15-2TT, University of Birmingham, May 1996
23. M. A. Stricker and M. Orengo. **Similarity of Color Images**. In Storage and Retrieval for Image and Video Databases (SPIE), pages 381–392, 1995.
24. M. Swain and D. Ballard. **Color Indexing**. International Journal of Computer Vision, 7(1):11–32, 1991.

25. G. Pass, R. Zabih, and J. Miller. **Comparing Images Using Color Coherence Vectors.** In Proceedings of the fourth ACM international conference on Multimedia, pages 65–73,1996.
26. Stehling,M.A. Nascimento, and A.X. Falcão. **A Compact and Efficient Image Retrieval Approach Based on Border/Interior Pixel Classification**. In Proceedings of the 11th ACM International. Conference on Information and Knowledge Management, pages 102–109,McLean, Virginia, USA, November 2002.
27. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih. **Image Indexing Using Color Correlograms.** In IEEE International Conference on Computer Vision and Pattern Recognition,pages 762–768, Puerto Rico, June 1997.
28. Han, J. and Kuang, K., **Fuzzy color histogram and its use in color image retrieval**. IEEE Trans. Image Process. v11 i8. 944-952.
29. Fuhui Long, Hongjiang Zhang, David D. Feng: **Fundamentals of Content-based Image retrieval**, in Multimedia Information Retrieval and Management - Technological Fundamentals and Applications, D. Feng, W.C. Siu, and H.J.Zhang. (ed.), Springer, 2002.
30. Y. Rubner, L.J. Guibas, and C. Tomasi, **"The Earth Mover's Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval,"** Proc. DARPA Image Understanding Workshop, pp. 661- 668, May 1997.
31. W.Y. Ma and B. Manjunath**, "NaTra: A Toolbox for Navigating Large Image Databases,"** Proc. IEEE Int'l Conf. Image Processing, pp. 568-571, 1997.
32. Jorma Laaksonen, Markus Koskela, Sami Laakso, and Erkki Oja. **Picsom – content-based image retrieval with self-organizing maps**. *Pattern Recognition Letters*, 21:1199–1207, 2000.
33. J.Z. Wang, G. Wiederhold, O. Firschein, and X.W. Sha, **"Content-Based Image Indexing and Searching Using Daubechies' Wavelets**", Int'l J. Digital Libraries, vol. 1, no. 4, pp. 311-328, 1998.Image Database,
34. http://wang.ist.psu.edu/docs/related.shtml [Last viewed March 25, 2008]
35. Linda G. Shapiro , George C. Stockman , "Computer Vision", Prentice-Hall, 2000

36. http://www.linuxtoday.com/news_story.php3?ltsn=2001-07-27-006-20-RV-SW [Last viewed April 28,2008]
37. N. Beckmann, *et al*, "The R*-tree: An efficient robust access method for points and rectangles," *ACM SIGMOD Int. Conf. on Management of Data*, Atlantic City, May 1990.
38. J. Vendrig, M. Worring, and A. W. M. Smeulders, "Filter image browsing: exploiting interaction in retrieval," Proc. Viusl'99: Information and Information System, 1999.
39. J. T. Robinson, "The k-d-B-tree: a search structure for large multidimensional dynamic indexes," *Proc. of SIGMOD Conference*, Ann Arbor, April 1981.
40. J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The grid file: an adaptable symmetric multikey file structure," ACM Trans. on Database Systems, pp. 38-71, March 1984.