CHAPTER I

1. Introduction

In many areas of commerce, government, academia, and hospitals, large collections of digital images are being created. Many of these collections are the product of digitizing existing collections of analogue photographs, diagrams, drawings, paintings, and prints. Usually, the only way of searching these collections was by keyword indexing, or simply by browsing. Digital images databases however, open the way to content-based searching. Recent years have seen a rapid increase in the size of digital image collections. Everyday imaging equipment generates Giga-bytes of images. Different surveys have estimated that world-wide 2,600 new images are created per second (equivalent to 80 billion per year) with an estimated 10 billion of which are available on the internet [1, 2, 3]. Finding the correct image has become an expensive problem. It is necessary to have these data organized so as to allow efficient browsing, searching, and retrieval. Image retrieval has been a very active research area since the 1970s, with the thrust from two major research communities; database management and computer vision. Image retrieval can be divided into text-based image retrieval (TBIR) and content-based image retrieval (CBIR) [1, 2, 3, 4, 5]. The text-based image retrieval technique first annotates the images by text, and then uses text-based database management systems to perform image retrieval. However, there exist two major difficulties, especially when the size of image collections is large. One is the vast amount of labour required in manual image annotation. The other difficulty is the subjectivity of human perception, that is, for the same image content different people may perceive it differently. The perception subjectivity and annotation impreciseness may cause unrecoverable mismatches in later retrieval processes. As processors become increasingly powerful, and memories become increasingly cheaper, the deployment of large image databases for a variety of applications have now become realizable. Databases of art works, satellite and medical imagery have been attracting more and more users in various professional fields for example, geography, medicine, architecture, advertising, design, fashion, and publishing. Effectively and efficiently accessing desired images from large and varied image databases is now a necessity

1.1 Content Based Image Retrieval

Content-based image retrieval (CBIR) is a technique which uses visual contents to search images from large scale image databases according to users' interest. It has been an active and fast advancing research area since the 1990s [1,2]. Content-based image retrieval uses the visual contents of an image such as color, shape, texture, and spatial layout to represent and index the image. [1,2,3] In typical content-based image retrieval systems(figure1.1), the visual contents of the images in the database are extracted and described by multi-dimensional feature vectors. The feature vectors of the images in the database form a feature database. To retrieve images, users provide the retrieval system with example images or sketched figures. The system then changes these examples into its internal representation of feature vectors. The similarities/distances between the feature vectors of the query example or sketch and those of the images in the database are then calculated and retrieval is performed with the aid of an indexing scheme. [6] The indexing scheme provides an efficient way to search for the image database. Recent retrieval systems have incorporated users' relevance feedback to modify the retrieval process in order to generate perceptually and semantically more meaningful retrieval results.

In CBIR, each image that is stored in the database has its features extracted and compared to the features of the query image. It involves two steps:

1. **Feature Extraction:** The first step in the process is extracting the low level feature and representing them.
2. **Similarity Measures:** The second step in the process is matching the query image with the different images present in the database according to their low level feature.

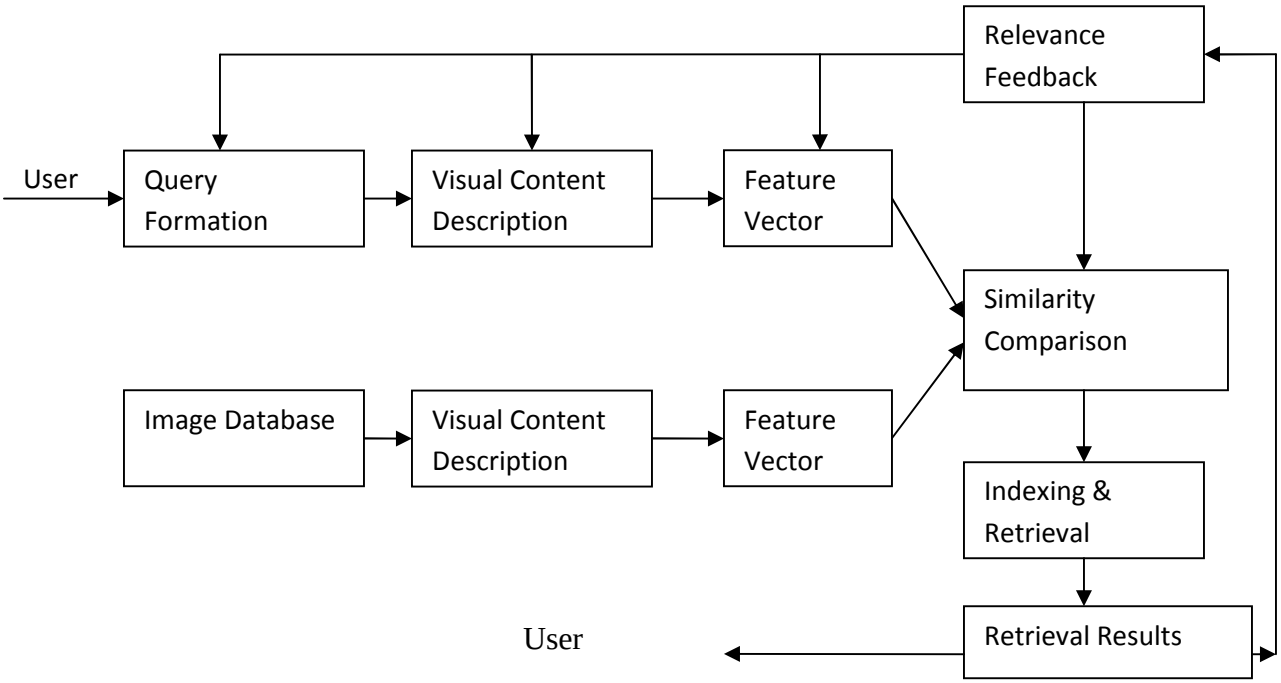


Figure 1.1: *Block diagram of Content based Image Retrieval*

1.1.1 Feature Extraction

Feature extraction is the basis of content-based image retrieval, and features can be classified as General Visual contents or Domain-specific Visual Contents [2,3,4]. General features are suitable for most applications and include color, texture, shape, color layout and shape layout features. Domain-specific features are only suitable for a narrow image domain. This research concentrates on general visual contents. Color, texture and shape are the most used features in content-based image retrieval. Many techniques are used to extract the feature of the images and represent them in the statistical form. The most commonly used technique to extract and represent the image feature in statistical form is Histogram.

1.1.2 Similarity Measures

Similarity measure technique is the second step in the process. After formulation of the low level feature of the image and extraction of the feature of the query image, the next step is to extract the features of the all the images present in the database and index them according to low level feature which makes searching fast and effective. Now the query image feature and database

image feature is matched or compared to retrieve the similar image according to low level content. Many similarity techniques are used to implement CBIR system. The most efficiently used technique is Minkowsi- Form Distance. Other techniques are Euclidian Distance, Quadratic Form Distance [6,7,8].

1.2 Motivation and Aims

Recent years have seen a rapid increase in the size of digital image collections in the internet and other systems. Everyday imaging equipment generates huge amount of images. The storage and retrieval of images is very difficult, which needs great amount of time and annotation technique to arrange the data with their name. Due to two major difficulties of manual image annotation and the subjectivity of human perception may cause unrecoverable mismatches in later retrieval processes. This leads to more efforts being required on content-based image retrieval. Using this technique, images are indexed by their own visual content, such as colour, texture or shape. CBIR has become more and more important with the advance of computer technology, and provides the answer to the two drawbacks of the Text based image Retrieval (TBIR) system mentioned above. It is important to stress that CBIR is not a replacement of, but rather a complementary component to TBIR. Only the integration of the two can result in satisfactory retrieval performance at present. The wavelet transform, a powerful tool in image processing and analysis will be used as the main tool for the research work.

1.3 Application of CBIR

Examples of CBIR applications are

- **Crime prevention:** Automatic face recognition systems, used by police forces.
- **Security Check:** Finger print or retina scanning for access privileges.
- **Medical Diagnosis:** Using CBIR in a medical database of medical images to aid diagnosis by identifying similar past cases.

- **Intellectual Property:** Trademark image registration, where a new candidate mark is compared with existing marks to ensure no risk of confusing property ownership.

1.4 CBIR System

Several CBIR systems currently exist, and are being constantly developed. Examples are

- ***QBIC*** or ***Query By Image Content*** was developed by IBM, Almaden Research Centre, to allow users to graphically pose and refine queries based on multiple visual properties such as colour, texture and shape. It supports queries based on input images, user-constructed sketches, and selected colour and texture patterns.
- ***VIR Image Engine*** by Virage Inc., like QBIC, enables image retrieval based on primitive attributes such as colour, texture and structure. It examines the pixels in the image and performs an analysis process, deriving image characterization features .
- ***VisualSEEK*** and ***WebSEEK*** were developed by the Department of Electrical Engineering, Columbia University. Both these systems support colour and spatial location matching as well as texture matching.
- ***NeTra*** was developed by the Department of Electrical and Computer Engineering, University of California. It supports colour, shape, spatial layout and texture matching, as well as image segmentation.
- ***MARS*** or ***Multimedia Analysis and Retrieval System*** was developed by the Beckman Institute for Advanced Science and Technology, University of Illinois. It supports colour, spatial layout, texture and shape matching.
- ***Viper*** or ***Visual Information Processing for Enhanced Retrieval*** was developed at the Computer Vision Group, University of Geneva. It supports colour and texture matching.

1.5 Thesis Organization

Apart from introduction part, the other information is arranged in the subsequent chapters. In Chapter 2, a literature study of the background of the thesis area will be studied. The literature study includes the background of content-based image retrieval; with different methods of the feature extraction of different low level features of images i.e. color, texture, and shape. Since the feature extraction is the main theme of this thesis, the different feature extraction techniques are presented in detail in Chapter 3, Chapter 4 and Chapter 5. In Chapter 3, detail analysis of color feature extraction techniques are discussed. In Chapter 4, a comprehensive comparison between several texture feature extraction techniques are conducted, and the best features extraction techniques is identified. The performance of the texture methods will be evaluated in terms of accuracy and computation time. Chapter 5 describes several shape feature extraction technique available in the literature. Chapter 6 introduces to the different comparison techniques used for the matching of the images from the database and the query images. Chapter 7 gives the detail of Implementation of the used model for the Contained Based Image Retrieval using color and texture feature with best results. Chapter 8 gives the detail of result and testing of the used model for the Contained Based Image Retrieval using color and texture feature with best results. Finally, in Chapter 9, the conclusions of the research are gathered together and various avenues for future works are presented.

CHAPTER II

2. Background and Problem Formulation

As a result of advances in the internet and new digital image sensor technologies, the volume of digital images produced by scientific, educational, medical, industrial and other applications available to users increased dramatically [1]. The difficulties faced by text-based image retrieval became more and more severe. In text-based image retrieval, the major two difficulties are (a) images were first annotated with text and then searched using a text-based approach from traditional database management systems and (b) Retrieval system requires manual annotation of images. Obviously, annotating images manually is a cumbersome and expensive task for large image databases, and is often subjective, context-sensitive and incomplete.

It requires more efforts on content-based image retrieval. Using this technique, images are indexed by their own visual content, such as color, texture or shape [7]. If we can make suitable system then it can remove the drawbacks of the Text-based image retrieval system as mentioned.

2.1 Literature Reviews

There has been a lot of research works dealt with Content based image retrieval system in last decades and various methods are proposed for retrieval of images from large databases from Internet and other database server. Retrieval can be done on the basis of different low level features such as color, shape, texture, spatial layout etc

2.1.1 CBIR System Architecture

A simple image retrieval system is suitable for the broad image domain. There are three databases in the system architecture, the image database, the feature vector database, and the text annotation database [6]. The image database contains the raw images for visual display purposes. The feature vector database stores the visual features extracted from the images. This is the information needed to support CBIR. Finally the text annotation database contains the keywords

and free-text descriptions of the images. The text-based retrieval is included in the system because only the integration of the two methods can result in satisfactory retrieval performance. However the concern of this thesis is only towards the feature extraction and similarities measure of images, which are the two important phases in a CBIR system.

2.1.2 Feature Extraction

Feature extraction is the one of the most important part of CBIR system. Visual content can be very general or domain specific. General visual content include color, texture, shape, spatial relationship, etc. Domain specific visual content, like human faces, is application dependent and may involve domain knowledge [7,8]. Semantic content is obtained either by textual annotation or by complex inference procedures based on visual content [6]. This thesis concentrates on feature extraction of color, shape, texture.

2.1.2.1 Color

Color is most extensively used visual content for image retrieval. Its three dimensional values make its discrimination potentiality superior to the single dimensional gray values of image. Color reflects the chromatic attributes of the image as it is captured with a sensor. Each pixel of the image can be represented as a point in a 3D color space. Commonly used color spaces are HSV, RGB, L*u*v, L*a*b, Munsell. [10]

The commonly used color feature representation are the Color Histogram, color coherence vector, color correlogram and color moments. Color histograms are amongst the most traditional technique for describing the low-level properties of an image [11]. Histogram intersection is usually used as the similarity measure for the color histogram. Finally the color coherence vector differs from the color histogram in that it manages to capture information about the distribution of the colors spatially within the image [12]. The detail study and analysis of color feature is presented in chapter 3.

2.1.2.2 Texture

Texture is another important property of images. Texture representation method can be classified in two parts: structural and statistical. Structural methods, including morphological operator and adjacency graph, describe texture by identifying structural primitives and their placement rules [8,13]. Statistical methods characterize texture by the statistical distribution of the image intensity. The different statistical methods are Fourier power spectra, co-occurrence matrices, shift-invariant principal component analysis (SPCA), Tamura feature, Wold decomposition, Markov random field, fractal model, and multi-resolution filtering technique such as Gabor and Wavelet transform. [8,13]. The detail study and analysis of color feature is presented in chapter 4.

2.1.2.3 Shape

Shape features involve all the properties that capture conspicuous geometric details in the image. The shape representations can be divided into two categories, boundary-based and region-based. The former uses only the outer boundary of the shape while the latter use the entire shape region [4,14]. The most successful representatives for these two categories are Fourier descriptor and moment invariants. The main idea of a Fourier descriptor is to use the Fourier transformed boundary as the shape feature. A modified Fourier descriptor which is robust to noise and invariant to geometric transformation is proposed by Rui *et al* [15]. The main idea of moment invariants is to use region-based moments which are invariant to transformations, as the shape feature. Hu identified seven such moments. Based on his work, many improved versions emerged, such as the Zernike moment descriptors proposed by Teague [15,16]. The rotational invariance nature of the Zernike moment descriptors make it very useful in overcoming the shortcomings associated with Hu's moments. Besides the Fourier descriptor and moments, some other recent work in shape representation and matching includes the finite element method (FEM), the turning function, and the wavelet descriptors. The FEM defines a stiffness matrix which describes how each point on the object is connected to the other points. The turning function method is useful in comparing both convex and concave polygons. Wavelet descriptors have desirable properties such as multi-resolution representation, invariance, uniqueness,

stability and spatial localization [9]. The shape feature extraction techniques are discussed in detail in chapter 5

2.1.2.4 Spatial Layout

Regions or objects with similar color and texture properties can be easily distinguished by imposing spatial constraints. For example, regions of blue sky and ocean may have similar color histograms, but their spatial locations in images are different. Therefore spatial locations or regions or the spatial relationship between multiple regions in an image is very useful for searching images [17].

The most widely used representation of spatial relationship is the 2D strings, it is constructed by projecting images along the x and y directions. In addition to the 2D string, spatial quad-tree and symbolic image are also used for spatial information representation.

2.1.3 Similarity Measures

Basically all systems use the assumption of equivalence of an image and its representation in feature space. These systems often use measurement systems such as the easily understandable Euclidean vector space model for measuring distances between a query image (represented by its features) and possible results representing all images as feature vectors in an n dimensional vector space. This is done, although metrics have been shown to not correspond well to human visual perception. Several other distance measures do exist for the vector space model such as the city block distance, the Mahalanobis distance or a simple histogram intersection. Still, the use of high dimensional feature spaces has shown to cause problems and great care needs to be taken with the choice of distance measurement to be chosen in order to retrieve meaningful results. These problems with a similarity definition in high dimensional feature spaces are also known as the curse of dimensionality and have also been discussed in the domain of medical imaging. Another approach is a probabilistic framework to measure the probability that an image is

relevant. A relationship between probabilistic image retrieval and vector space distance measures is given in [6, 8, 18].

2.1.4 Multilevel Image Indexing

Multidimensional indexing is needed in content-based image retrieval in order to make the system truly scalable to large size image collections.[19] Similarity retrieval on multiple attributes (as opposed to exact matching) is also an important issue for multimedia systems, as most of the queries are searching for the nearest matches rather than exact matches. Many different approaches for multidimensional indexing can be found, such as the many branches of the famous tree-based algorithms. Clustering and neural nets, widely used in pattern recognition, are also promising indexing techniques. [6] The history of multidimensional indexing techniques can be traced back to the middle of 1970s, when cell methods, quad-tree, and k-d tree were first introduced. Guttman proposed the R-tree indexing structure [20]. Based on his work, many other variants of the R-tree were developed. There are two main types of multidimensional indexing method: Point Access Methods and Spatial Access Methods. The Point Access Method is only for multidimensional data points whereas the Spatial Access Methods are designed for data not only represented by points but also as multidimensional spatial regions.

CHAPTER III

3. Color Feature Extraction and Analysis

3.1 Definition

One of the most important features that make possible the recognition of images by humans is colour. Colour is a property that depends on the reflection of light to the eye and the processing of that information in the brain. Colour is used everyday by us to tell the difference between objects, places, and the time of day.

3.1.1 Color Space

Usually colours[21] are defined in three dimensional colour spaces. These could either be *RGB* (Red, Green, and Blue), *Munsell*, *CIE L*a*b**, *CIE L*u*v**, *HSV* (Hue, Saturation, and Value) or *HSB* (Hue, Saturation, and Brightness)[21]. The last two Color Spaces are dependent on the human perception of hue, saturation, and brightness. The Munsell color system is a color space that specifies colors based on three color dimensions, hue, value (lightness), and chroma (color purity or colorfulness). It was created by Professor Albert H. Munsell in the first decade of the 20th century. The Color spaces are described in detail in consequent chapters.

Most image formats such as *JPEG*, *BMP*, *GIF*, use the RGB colour space to store information. The RGB colour space is defined as a unit cube with red, green, and blue axes. Thus, a vector with three co-ordinates represents the colour in this space. When all three coordinates are set to zero the colour perceived is black. When all three coordinates are set to 1 the colour perceived is white. The other colour spaces operate in a similar fashion but with a different perception. There is no agreement on which is the best. However, one of the desirable characteristics of an appropriate color space for image retrieval is its *uniformity*. Uniformity means that two color pairs that are equal in similarity distance in a color space are perceived as equal by viewers. In other words, the measured proximity among the colors must be directly related to the psychological similarity among them [30].

a) RGB Color Space

RGB space is a widely used color space for image display. It is composed of three color components *red*, *green*, and *blue*. These components are called "*additive primaries*" since a color in RGB space is produced by adding them together [21]. Each color point within the bounds of the cube can be represented as triple (R, G, B), where values for R, G, and B are assigned in the range from 0 to 1.

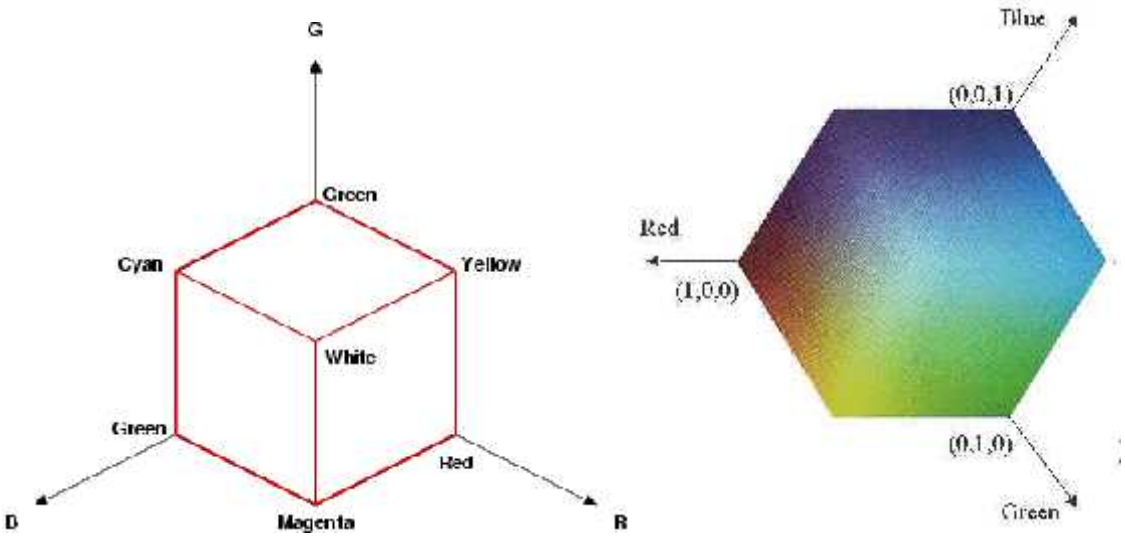


Figure 3.1: *RGB color space*

b) CMY Color Space

In contrast, CMY space [21] is a color space primarily used for printing. The three color components are *cyan*, *magenta*, and *yellow*. These three components are called "*subtractive primaries*" since a color in CMY space is produced through light absorption. Both RGB and CMY space are device-dependent and perceptually non-uniform.

c) CIE Color Space

The CIE L*a*b* and CIE L*u*v* [21] spaces are device independent and considered to be perceptually uniform. They consist of a luminance or *lightness* component (*L*) and two *chromatic*

components *a* and *b* or *u* and *v*. CIE L*a*b* is designed to deal with subtractive colorant mixtures, while CIE L*u*v* is designed to deal with additive colorant mixtures. The transformation of RGB space to CIE L*u*v* or CIE L*a*b* space can be found in [21].

d) HSV Color Space

In HSV (or HSL, or HSB) space is widely used in computer graphics and is a more intuitive way of describing color. The three color components are *hue*, *saturation* (lightness) and *value* (*brightness*) [21]. The hue is invariant to the changes in illumination and camera direction and hence more suited to object retrieval. RGB coordinates can be easily translated to the HSV (or HLS, or HSB) coordinates by a simple formula. The opponent color space uses the opponent color axes (*R-G*, *2B-R-G*, *R+G+B*). This representation has the advantage of isolating the brightness information on the third axis. With this solution, the first two chromaticity axes, which are invariant to the changes in illumination intensity and shadows, can be down-sampled since humans are more sensitive to brightness than they are to chromatic information.

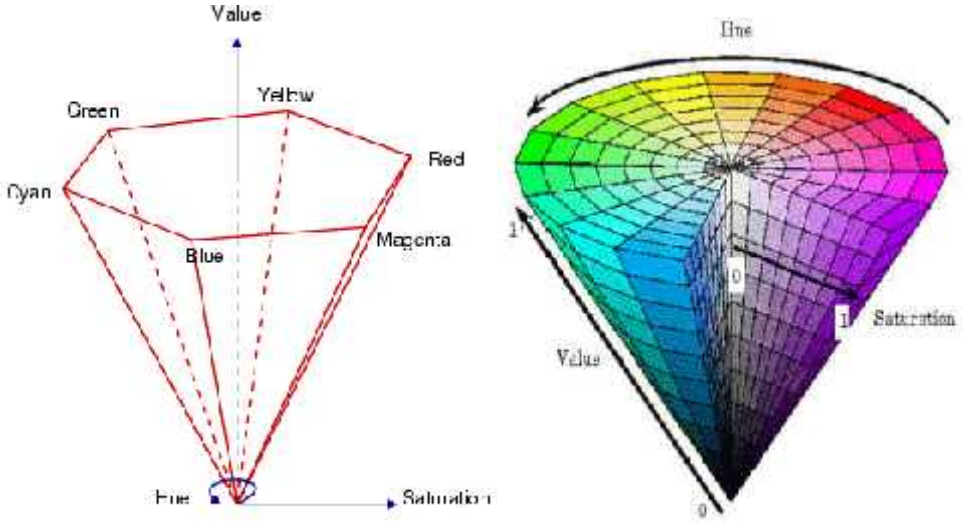


Figure 3.2: *HSV color space*

3.2 Methods of Representation

There many way to describe the color feature of the image. Some of the most broadly used are introduced like color descriptors: the color histogram, color coherence vector, color correlogram, and color moments.

3.2.1 Color Histogram

The main method of representing colour information of images in CBIR systems is through colour histograms. A colour histogram [12,22] is a type of bar graph, where each bar represents a particular colour of the colour space being used. The popularity of Color Histogram is due to Color histograms are computationally trivial to compute, Small changes in camera viewpoint tend not to effect color histograms and different objects often have distinctive color histograms. The bars in a colour histogram are referred to as bins and they represent the x-axis. The number of bins depends on the number of colours there are in an image. The y-axis denotes the number of pixels there are in each bin. In other words how many pixels in an image are of a particular colour.

In histogram representation of Images, all the images are scaled to contain the same number of pixels M. We discretize the color space of the image such that there are n distinct (discretized) colors. A color histogram H is a vector (h_1; h_2; : : : ; h_n), in which each bucket h_j contains the number of pixels of color j in the image. Typically images are represented in the RGB colors pace, and a few of the most significant bits are used from each color channel. For a given image I, the color histogram H_I is a compact summary of the image. A database of images can be queried to find the most similar image to I, and can return the image I' with the most similar color histogram $H_{I'}$. Typically color histograms are compared using the sum of squared differences (L2-distance) or the sum of absolute value of differences (L1-distance). So the most similar image to I would be the image I' minimizing

$$\|H_1 - H_1'\| = \sum_{j=1}^{n} f H_I[j] - H_{I'}[j] \hat{A}^2$$

for the L2-distance[22,23], or

$$| H_I \; Z \; H_{I'} | X \sum_{j X 1}^{n} | H_I [j] \; Z \; H_{I'} [j] |$$

for the L1-distance.

An example of a colour histogram in the HSV colour space can be seen with the following image:

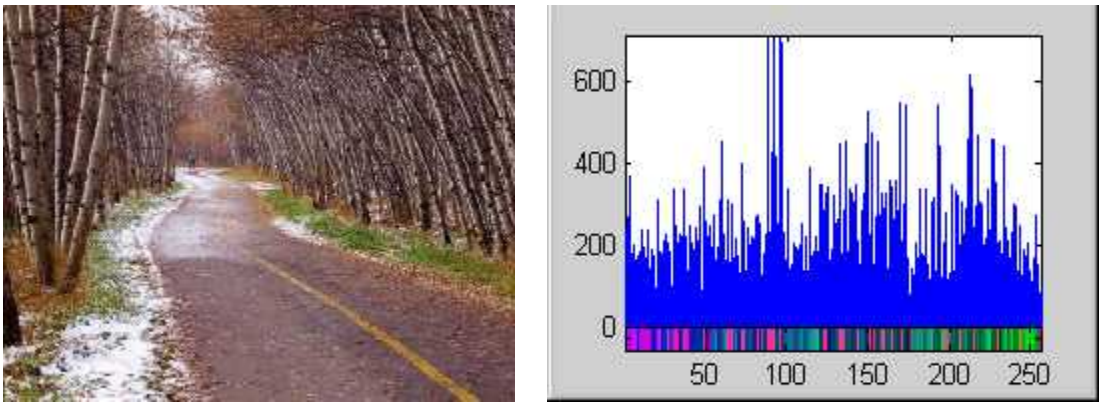


Figure 3.3: *Sample Image and its Corresponding Histogram*

To view a histogram numerically one has to look at the colour map or the numeric representation of each bin. Each bin contains different numerical values for the different component present in the image. Here in our case, the Image contains the image contains the values for Hue(H), Saturation(S), and Values(V) which are the building block of the HSV color Space.

Colour Map (x-axis)			***Number of Pixels per Bin (y-axis)***
H	***S***	***V***	
0.9922	0.9882	0.9961	106
0.9569	0.9569	0.9882	242
0.9725	0.9647	0.9765	273
0.9176	0.9137	0.9569	372
0.9098	0.8980	0.9176	185
0.9569	0.9255	0.9412	204
0.9020	0.8627	0.8980	135
0.9020	0.8431	0.8510	166
0.9098	0.8196	0.8078	179
0.8549	0.8510	0.8941	188
0.8235	0.8235	0.8941	241
0.8471	0.8353	0.8549	104
0.8353	0.7961	0.8392	198
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Table 3.1: *Colour Map and Number of pixels for the Image of figure 3.3*

As one can see from the colour map each row represents the colour of a bin. The row is composed of the three coordinates of the colour space. The first coordinate represents hue, the second saturation, and the third, value, thereby giving HSV. The percentages of each of these coordinates are what make up the colour of a bin. Also one can see the corresponding pixel numbers for each bin, which are denoted by the blue lines in the histogram [20].

Quantization in terms of colour histograms refers to the process of reducing the number of bins by taking colours that are very similar to each other and putting them in the same bin. By default the maximum number of bins one can obtain using the histogram function in MatLab is 256. For the purpose of saving time when trying to compare colour histograms, one can quantize the number of bins. Obviously quantization reduces the information regarding the content of images but as was mentioned this is the tradeoff when one wants to reduce processing time.

There are two types of colour histograms, Global colour histograms (***GCH***s) and Local colour histograms (***LCH***s). A GCH represents one whole image with a single colour histogram. An LCH divides an image into fixed blocks and takes the colour histogram of each of those blocks [1,22]. LCHs contain more information about an image but are computationally expensive when comparing images. "The GCH is the traditional method for colour based image retrieval. However, it does not include information concerning the colour distribution of the regions" [1,2,3,22] of an image. Thus when comparing GCHs one might not always get a proper result in terms of similarity of images.

When an image database contains a large number of images, histogram comparison will saturate the discrimination. To solve this problem, the *joint histogram* technique is introduced [22,23]. In addition, color histogram does not take the spatial information of pixels into consideration, thus very different images can have similar color distributions. This problem becomes especially acute for large scale databases. To increase discrimination power, several improvements have been proposed to incorporate spatial information. A simple approach is to divide an image into sub-areas and calculate a histogram for each of those sub-areas. As introduced above, the division can be as simple as a rectangular partition, or as complex as a region or even object segmentation. Increasing the number of sub-areas increases the information about location, but also increases the memory and computational time.

3.2.2 Color Coherence Vector

In [26] a different way of incorporating spatial information into the color histogram, C*olor coherence vectors* (*CCV*), was proposed. Each histogram bin is partitioned into two types, i.e., coherent, if it belongs to a large uniformly-colored region, or incoherent, if it does not.

Figure 3.4: *Two image with similar Color histogram*

For example, the images shown in figure 3.4, two images have similar color histograms, despite their rather different appearances. The color red appears in both images in approximately the same quantities. In the left image the red pixels (from the flowers) are widely scattered, while in the right image the red pixels (from the golfer's shirt) form a single coherent region. The coherence measure classifies pixels as either coherent or incoherent. Coherent pixels are a part of some sizable contiguous region, while incoherent pixels are not. A *color coherence vector* represents this classification for each color in the image. CCV's prevent coherent pixels in one image from matching incoherent pixels in another. This allows fine distinctions that cannot be made with color
Histograms[13].

Let α_i denote the number of coherent pixels in the i^{th} color bin and β_i denote the number of incoherent pixels in an image. Then, the CCV of the image is defined as the vector $<(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_N, \beta_N)>$. Note that $<\alpha_1+\beta_1, \alpha_2+\beta_2, \ldots, \alpha_N+\beta_N>$ is the color histogram of the image. Due to its additional spatial information, it has been shown that CCV provides better retrieval results

than the color histogram, especially for those images which have either mostly uniform color or mostly texture regions. In addition, for both the color histogram and color coherence vector representation, the HSV color space provides better results than CIE L*u*v* and CIE L*a*b* spaces [2,13,21].

3.2.3 Color Correlogram

A color correlogram [22] (henceforth correlogram) expresses how the spatial correlation of pairs of colors changes with distance. Informally, a correlogram for an image is a table indexed by color pairs, where the k-th entry for row (i, j) specifies the probability of finding a pixel of color j at a distance k from a pixel of color i in this image. Here Ic is chosen from a set of distance values D. The *color correlogram* [23] was proposed to characterize not only the color distributions of pixels, but also the spatial correlation of pairs of colors. The first and the second dimension of the three-dimensional histogram are the colors of any pixel pair and the third dimension is their spatial distance. A color correlogram is a table indexed by color pairs, where the *k*-th entry for (*i, j*) specifies the probability of finding a pixel of color *j* at a distance *k* from a pixel of color *i* in the image. Let *I* represent the entire set of image pixels and *Ic(i)* represent the set of pixels whose colors are *c(i).* Then, the color correlogram is defined as

$$¥_{i,j}^{(k)} X \underset{p_1}{Pr}\left[I_{c(i)}, p_2 \in I \;/\; | p1 Z p2 |X k \right]$$

where *i, j* Č {*1, 2, …, N*}, *k*Č {*1, 2, …, d*}, and | *p1 – p2* | is the distance between pixels *p1* and *p2*. If we consider all the possible combinations of color pairs the size of the color correlogram will be very large (*O(N*^{2}*d)*), therefore a simplified version of the feature called the *color autocorrelogram* is often used instead. The color autocorrelogram only captures the spatial correlation between identical colors and thus reduces the dimension to *O(Nd).*

Compared to the color histogram and CCV, the color autocorrelogram provides the best retrieval results, but is also the most computational expensive due to its high dimensionality.

3.2.4 Color Moments

Color moments [24] have been successfully used in many retrieval systems (like *QBIC*) [2,3], especially when the image contains just the object. The *first order* (*mean*), the *second* (*variance*) and the *third order* (*skewness*) color moments have been proved to be efficient and effective in representing color distributions of images. Mathematically, the first three moments are defined as:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} f_{ij}$$

$$\sigma_i = \left(\frac{1}{N}\sum_{j=1}^{N}\left(f_{ij}-\mu_i\right)^2\right)^{\frac{1}{2}}$$

$$s_i = \left(\frac{1}{N}\sum_{j=1}^{N}\left(f_{ij}-\mu_i\right)^3\right)^{\frac{1}{3}}$$

where f_{ij} is the value of the i-th color component of the image pixel j, and N is the number of pixels in the image. Usually the color moment performs better if it is defined by both the L*u*v* and L*a*b* color spaces as opposed to solely by the HSV space Using the additional third-order moment improves the overall retrieval performance compared to using only the first and second order moments. However, this third-order moment sometimes makes the feature representation more sensitive to scene changes and thus may decrease the performance.

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. Due to this compactness, it may also lower the discrimination power. Usually, color moments can be used as the first pass to narrow down the search space before other sophisticated color features are used for retrieval.

CHAPTER IV

4. Texture Feature Extraction and Analysis

The efficient way of the feature representation of the image is in the form of texture of the image.

4.1 Definition

Texture [17,27] is that innate property of all surfaces that describes visual patterns, each having properties of homogeneity. It contains important information about the structural arrangement of the surface, such as; clouds, leaves, bricks, fabric, etc. It also describes the relationship of the surface to the surrounding environment. In short, it is a feature that describes the distinctive physical composition of a surface.

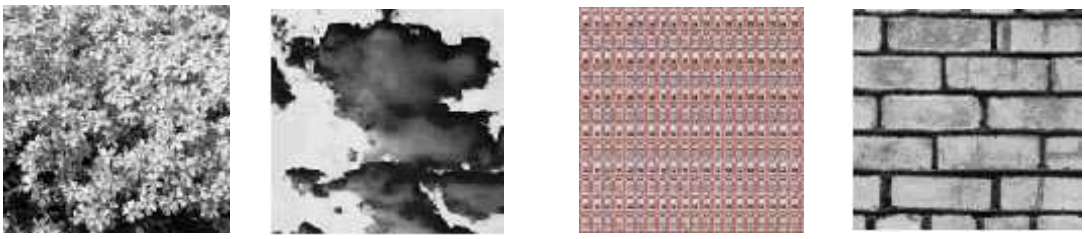


Figure 4.1: *Example of Textures*

Texture properties include:

- Coarseness
- Contrast
- Directionality
- Line-likeness
- Regularity
- Roughness

Texture is one of the most important defining features of an image. It is characterized by the spatial distribution of gray levels in a neighborhood. In order to capture the spatial dependence of gray-level values, which contribute to the perception of texture, a two-dimensional dependence texture analysis matrix is taken into consideration. This two-dimensional matrix is obtained by decoding the image file; jpeg, bmp, etc.

4.2 Methods of Representation

There are three principal approaches used to describe texture; statistical, structural and spectral

- Statistical techniques characterize textures using the statistical properties of the grey levels of the points/pixels comprising a surface image. Typically, these properties are computed using: the grey level co-occurrence matrix of the surface, or the wavelet transformation of the surface.
- Structural techniques characterize textures as being composed of simple primitive structures called “texels” (or texture elements). These are arranged regularly on a surface according to some surface arrangement rules.
- Spectral techniques are based on properties of the Fourier spectrum and describe global periodicity of the grey levels of a surface by identifying high-energy peaks in the Fourier spectrum [13,27].

For optimum classification purposes, the statistical techniques of characterization are used. This is because it is these techniques that result in computing texture properties. The most popular statistical representations of texture are:

- Co-occurrence Matrix
- Tamura Texture
- Simultaneous Auto-Regressive (SAR) Model
- Wavelet Transform

4.2.1 Tamura Texture

By observing psychological studies in the human visual perception, Tamura explored the texture representation using computational approximations to the six main texture features of: coarseness, contrast, directionality, likeliness, regularity and roughness. In the most cases, only the first three Tamura's features are used for the CBIR. The features capture the high-level perceptual attributes of a texture well and are useful for image browsing. Each of these texture features are approximately computed using algorithms.

- *Coarseness* is the measure of granularity of an image, or average size of regions that have the same intensity.
- *Contrast* is the measure of vividness of the texture pattern. Therefore, the bigger the blocks that makes up the image, the higher the contrast. It is affected by the use of varying black and white intensities.
- *Directionality* is the measure of directions of the grey values within the image.

4.2.1.1 Coarseness

Coarseness is a measure of the granularity of the texture. It is defined as the distances of notable variations of grey levels of the image. In another way, it can be defined as the differences between the average signals for the non-overlapping windows of different size. To calculate the coarseness, moving averages $A_k(x,y)$ are computed first using $2^k \times 2^k$ (k = 0, 1, …, 5) size windows at each pixel (x, y) i. e.

$$A_k f x , y \mathrm{AX} \sum_{i \mathrm{X} x \mathrm{Z} 2^{k \mathrm{Z} 1}}^{x \Gamma 2^{k \mathrm{Z} 1} \mathrm{Z} 1} \sum_{j \mathrm{X} y \mathrm{Z} 2^{k \mathrm{Z} 1}}^{y \Gamma 2^{k \mathrm{Z} 1} \mathrm{Z} 1} g (i , j) / 2^{2k}$$

Then, the differences between pairs of non-overlapping moving averages in the horizontal and vertical directions for each pixel are computed, i.e.,

$$E_{k,h}(x, y) \mathrm{X} \left| A_k (x \Gamma 2^{k \mathrm{Z} 1}, y) \mathrm{Z} A_k (x \mathrm{Z} 2^{k \mathrm{Z} 1}, y) \right|$$

$$E_{k,v}(x, y) \mathrm{X} \left| A_k (x, y \Gamma 2^{k \mathrm{Z} 1}) \mathrm{Z} A_k (x, y \mathrm{Z} 2^{k \mathrm{Z} 1}) \right|$$

After that, the value of k that maximizes E in either direction is used to set the best size for each pixel, i.e.,

$$S_{best}(x, y) \mathrm{X} 2^k$$

The coarseness is then computed by averaging S_{best} over the entire image, i.e.,

$$F_{crs} \; \mathrm{X} \; \frac{1}{m \mid n} \sum_{i\mathrm{X}1}^{m} \sum_{j\mathrm{X}1}^{n} S_{best}(i, j)$$

Instead of taking the average of S_{best} an improved version of the coarseness feature can be obtained by using a histogram to characterize the distribution of S_{best}. Compared with using a single value to represent coarseness, using histogram-based coarseness representation can greatly increase the retrieval performance. This modification makes the feature capable of dealing with an image or region which has multiple texture properties, and thus is more useful to image retrieval applications.

4.2.1.2 Contrast

The contrast feature can be evaluated by finding the vary of gray level of the image **g** and to what extent their distribution is biased to black or white

$$F_{con} \; \mathrm{X} \; \frac{\pm}{\sim_{4}^{1/4}}$$

Where the kurtosis $\alpha_4 = \mu_4/\sigma^4$, μ4 is the fourth moment about the mean, and σ^2 is the variance. This formula can be used for both the entire image and a region of the image.

4.2.1.3 Directionality

Directionality of texture image distribution is identified by frequency distribution of oriented local edges against their directional angles. The edge strength e(x,y) and the directional angle a(x,y) are computed using approximate pixel-wise derivatives computed by the Soble edge detector in the 3×3 moving windows.

$$\begin{vmatrix} \mathrm{Z}1 & 0 & 1 \\ \mathrm{Z}1 & 0 & 1 \\ \mathrm{Z}1 & 0 & 1 \end{vmatrix} \text{ and } \begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ \mathrm{Z}1 & \mathrm{Z}1 & \mathrm{Z}1 \end{vmatrix}$$ and a gradient vector at each pixel is computed.

The magnitude and angle of this vector are defined as

$|\zeta G| \mathrm{X}(|\zeta_H|\Gamma|\zeta_V|)/2$

$^{-}\ \mathrm{X}\tan^{Z1}(\zeta_V/\zeta_H)\Gamma\circledR/2$

where ΔH and ΔV are the horizontal and vertical differences of the convolution. Then, by quantizing θ and counting the pixels with the corresponding magnitude |ΔG| larger than a threshold, a histogram of θ, denoted as *HD*, can be constructed. This histogram will exhibit strong peaks for highly directional images and will be relatively flat for images without strong orientation. The entire histogram is then summarized to obtain an overall directionality measure based on the sharpness of the peaks:

$$F_{dir}\ \mathrm{X}\ \sum_{p\ \ \square\ \ w_p}^{n_p}(\square\ \mathrm{Z}\square_p)^2 H_D(\square)$$

4.2.1.4 Linelikeness (F_{lin})

Linelikeness[17] is defined as an average coincidence of the coded directional angles in the pairs of pixels separated by a distance *d* along the edge direction in every pixel

4.2.1.5 Regularity

Regularity[17] can be obtained by using following formula $F_{\mathrm{reg}}=1-r(\mathrm{S}_{crs}+S_{\mathrm{con}}+S_{\mathrm{dir}}+S_{\mathrm{lin}})$ where *r* is a normalizing factor and each *S* is the standard deviation of the corresponding feature *F* in each sub image the texture is partitioned into roughness.

4.2.1.6 Roughness

The roughness [13] can be find by adding the energies obtained from the coarseness computation and contrast computation.

$F_{\mathrm{rgh}}=F_{\mathrm{crs}}+F_{\mathrm{con}}$

4.2.2 Co-occurrence Matrix

Tamura features of image only explain the only high level perceptual attributes of the texture. But they are not very effective in finer image discrimination. More intricate texture feature based on second order signal statistics i.e. Co-occurrence matrix was proposed. Originally proposed by

R.M. Haralick, the co-occurrence matrix representation of texture features explores the grey level spatial dependence of texture. A mathematical definition of the co-occurrence matrix is as follows [27,30]:

- Given a position operator ***P(i,j)***,
- let ***A*** be an n x n matrix
- whose element ***A[i][j]*** is the number of times that points with grey level (intensity) ***g[i]*** occur, in the position specified by ***P***, relative to points with grey level ***g[j]***.
- Let ***C*** be the ***n* x *n*** matrix that is produced by dividing ***A*** with the total number of point pairs that satisfy ***P***. ***C[i][j]*** is a measure of the joint probability that a pair of points satisfying ***P*** will have values ***g[i]***, ***g[j]***.
- ***C*** is called a co-occurrence matrix defined by ***P***.

Examples for the operator ***P*** are: "***i*** above ***j***", or "***i*** one position to the right and two below ***j***", etc. [27]

This can also be illustrated as follows… Let *t* be a translation, then a co-occurrence matrix C_t of a region is defined for every grey-level *(a, b)* by [27]:

$$C_t(a,b) \text{ X } card\{(s, s\,\Gamma\, t) \quad R^2 \mid A[s] \text{ X } a, A[s\,\Gamma\, t] \text{ X } b\}$$

Here, $C_t(a, b)$ is the number of site-couples, denoted by *(s, s + t)* that are separated by a translation vector *t*, with *a* being the grey-level of *s*, and *b* being the grey-level of *s + t*.

For example; with an 8 grey-level image representation and a vector *t* that considers only one neighbour, It is found:

1	2	1	3	4
2	3	1	2	4
3	3	2	1	1

Figure 4.2: *Co- occurrence Matrix of Image*

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	0	0	0	0	0
2	0	1	0	2	0	0	0	0
3	0	0	1	1	0	0	0	0
4	0	1	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

***Figure 4.3**: Classical Co-occurrence matrix*

At first the co-occurrence matrix is constructed, based on the orientation and distance between image pixels. Then meaningful statistics are extracted from the matrix as the texture representation [13,30]. Haralick proposed the following texture features:

1. Angular Second Moment
2. Contrast
3. Correlation
4. Variance
5. Inverse Second Differential Moment
6. Sum Average
7. Sum Variance
8. Sum Entropy
9. Entropy
10. Difference Variance
11. Difference Entropy
12. Measure of Correlation 1
13. Measure of Correlation 2
14. Local Mean

Hence, for each Haralick texture feature, a co-occurrence matrix is obtained. These co-occurrence matrices represent the spatial distribution and the dependence of the grey levels within a local area. Each ***(i,j)*** th entry in the matrices, represents the probability of going from one

pixel with a grey level of '***i***' to another with a grey level of '***j***' under a predefined distance and angle. From these matrices, sets of statistical measures are computed, called feature vectors [8].

4.2.3 Wavelet Transform

Textures can be modeled as quasi-periodic patterns[9,15,27] with spatial/frequency representation. The wavelet transform transforms the image into a multi-scale representation with both spatial and frequency characteristics. This allows for effective multi-scale image analysis with lower computational cost. According to this transformation, a function, which can represent an image, a curve, signal etc., can be described in terms of a coarse level description in addition to others with details that range from broad to narrow scales.

Unlike the usage of sine functions to represent signals in Fourier transforms, in wavelet transform, Functions known as wavelets are used. Wavelets are finite in time, yet the average value of a wavelet is zero. In a sense, a wavelet is a waveform that is bounded in both frequency and duration. While the Fourier transform converts a signal into a continuous series of sine waves, each of which is of constant frequency and amplitude and of infinite duration, most real-world signals (such as music or images) have a finite duration and abrupt changes in frequency. This accounts for the efficiency of wavelet transforms. This is because wavelet transforms convert a signal into a series of wavelets, which can be stored more efficiently due to finite time, and can be constructed with rough edges, thereby better approximating real-world signals.

Examples of wavelets are Coiflet, Morlet, Mexican Hat, Haar and Daubechies. Of these, Haar is the simplest and most widely used, while Daubechies have fractal structures and are vital for current wavelet applications [1]. These two are outlined below:

4.2.3.1 Haar Wavelet

The Haar wavelet [28], is an odd rectangular pulse pair, is the simplest and oldest orthonormal wavelet with compact support, and can be used for illustration purposes. Starting from the finer scales, the basic wavelet is translated by increments equal to its width, so that the complete set of wavelets at any scale completely covers the interval. The basic wavelet is progressively stretched (increased in scale) by powers of two. As the basic wavelet is scaled up by powers of two, its amplitude is scaled down by powers of 2, to maintain orthonormality. The result of this is a set of orthonormal basis functions. The scaling function completes the interval in the coarsest scale. Haar wavelet can be denoted by following graphs [28].

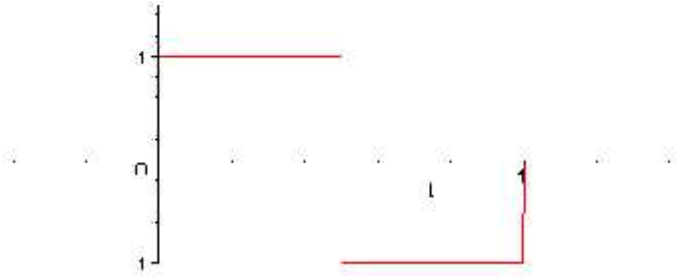


Figure 4.4: *Haar Wavelet Example*

4.2.3.2 Daubechies Wavelet

This wavelet is naamed after Ingrid Daubechies, the Daubechies wavelets are a family of orthogonal wavelets defining a discrete wavelet transform and characterized by a maximal number of vanishing moments for some given support. With each wavelet type of this class, there is a scaling function (also called father wavelet) which generates an orthogonal multiresolution analysis [29].

The Daubechies wavelet family is defined as [29]

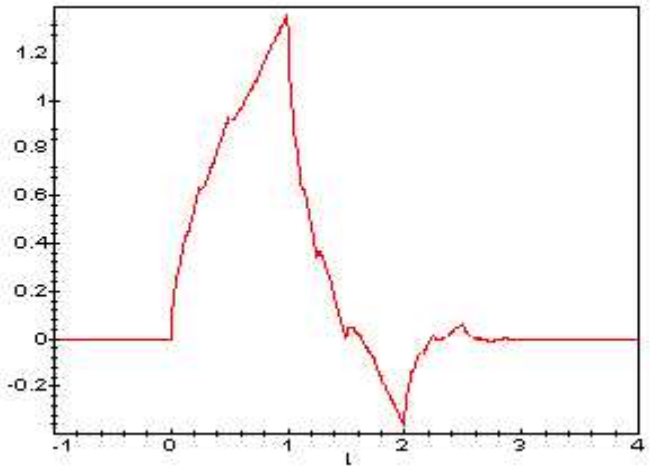


Figure 4.5: *Daubechies Wavelet Example*

4.2.4 Gabor Filter Feature

The *Gabor filter* has been widely used to extract image features, especially texture features [1]. It is optimal in terms of minimizing the joint uncertainty in space and frequency, and is often used as an orientation and scale tunable edge and line (bar) detector. There have been many approaches proposed to characterize textures of images based on Gabor filters. The basic idea of using Gabor filters to extract texture features is as follows.

A two dimensional Gabor function $g(x, y)$ is defined as:

$$g(x,y) \mathrm{X} \frac{1}{2\circledR \pm_X \pm_y} \exp\ \mathrm{Z} \frac{1}{2}\ \frac{x^2}{\pm_X^2} \Gamma \frac{y^2}{\pm_y^2}\ \ \Gamma 2\circledR\ jWx$$

where, σ_x and σ_y are the standard deviations of the Guassian envelopes along the x and y direction.

Then a set of Gabor filters can be obtained by appropriate dilations and rotations of *g(x,y):*

$$g_{mn}(x,y) = a^{-m}\, g(x',y')$$

$$x' = a^{-m}(x\cos\theta + y\sin\theta)$$

$$y' = a^{-m}(-x\sin\theta + y\cos\theta)$$

where $a > 1$, $\theta = n\pi/K$, $n = 0, 1, \ldots, K-1$, and $m = 0, 1, \ldots, S-1$. K and S are the number of orientations and scales. The scale factor $a-m$ is to ensure that energy is independent of m. Given an image $I(x, y)$, its Gabor transform is defined as:

$$W_{mn} = \int I(x,y)\, g^{*}_{mn}(x-x_1, y-y_1)dx_1dy_1$$

where * indicates the complex conjugate. Then the mean μ_{mn} and the standard deviation σ_{mn} of the magnitude of $W_{mn}(x, y)$, i.e., $f = [\mu_{00}, \sigma_{00,\text{-----}}, \mu_{mn}, \sigma_{mn}, \Lambda, \mu_{s\text{-}1k\text{-}1}, \sigma_{s\text{-}1k\text{-}1]}$ can be used to represent the texture feature of a homogenous texture region.

CHAPTER V

5 Shape Feature Extraction and Analysis

5.1 Definition

Shape[12,31] may be defined as the characteristic surface configuration of an object; an outline or contour. It permits an object to be distinguished from its surroundings by its outline. Shape representations can be generally divided into two categories:

- Boundary-based, and
- Region-based.

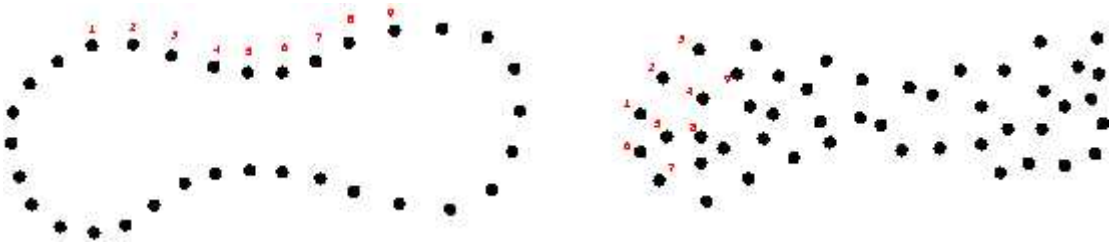


Figure 5.1: *Boundary-based & Region-based shape representation.*

Boundary-based shape representation only uses the outer boundary of the shape. This is done by describing the considered region using its external characteristics; i.e., the pixels along the object boundary. Region-based shape representation uses the entire shape region by describing the considered region using its internal characteristics; i.e., the pixels contained in that region.

5.2 Methods of Representation

For representing shape features mathematically, Following method are used [4,26]:

Boundary-based:

- Polygonal Models, boundary partitioning
- Fourier Descriptors
- Splines, higher order constructs
- Curvature Models

Region-based:

- Superquadrics
- Fourier Descriptors
- Implicit Polynomials
- Blum's skeletons

The most successful representations for shape categories are Fourier Descriptor and Moment Invariants[4,26]:

- The main idea of Fourier Descriptor is to use the Fourier transformed boundary as the shape feature.
- The main idea of Moment invariants is to use region-based moments, which are invariant to transformations as the shape feature.

5.2.1 Fourier Descriptor

Fourier descriptors describe the shape of an object with the Fourier transform of its boundary. Again, consider the contour of a 2D object as a closed sequence of successive boundary pixels (x_s, y_s), where $0 \le s \le N\text{-}1$ and N is the total number of pixels on the boundary. Then three types of contour representations, i.e., *curvature*, *centroid distance*, and *complex coordinate function*, can be defined.

The curvature $K(s)$ at a point s along the contour is defined as the rate of change in tangent direction of the contour, i.e.,

$$K(s) = \frac{d}{ds}(s)$$

where $\theta(s)$ is the turning function of the contour, The centroid distance is defined as the distance function between boundary pixels and the centroid (x_c, y_c) of the object:

$$R(s) = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2}$$

The complex coordinate is obtained by simply representing the coordinates of the boundary pixels as complex numbers:

$$Z(s) = (x_s - x_c) + j(y_s - y_c)$$

The Fourier transforms of these three types of contour representations generate three sets of complex coefficients, representing the shape of an object in the frequency domain. Lower frequency coefficients describe the general shape property, while higher frequency coefficients reflect shape details. To achieve rotation invariance (i.e., contour encoding is irrelevant to the choice of the reference point), only the amplitudes of the complex coefficients are used and the phase components are discarded. To achieve scale invariance, the amplitudes of the coefficients are divided by the amplitude of DC component or the first non-zero coefficient. The translation invariance is obtained directly from the contour representation.

5.2.2 Moment Invariants

Classical shape representation uses a set of *moment invariants* [1,2]. If the object *R* is represented as a binary image, then the central moments of order *p*+*q* for the shape of object *R* are defined as:

$$\mu_{p,q} = \sum_{(x,y)\ R} (x - x_c)^p (y - y_c)^q$$

where (*xc, yc*) is the center of object. This central moment can be normalized to be scale invariant

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{\gamma}}, \quad \gamma = \frac{p+q}{2} + 1$$

CHAPTER VI

6. Similarity Measures

For retrieval of the image from the large database, Content-based image retrieval calculates the visual similarities between the query image and images in the database instead of exact matching. The retrieval result is not a single image but a list of images ranked by their similarities with the query image. Many similarities measures technique have been developed in recent years. Different similarities/distance measures will affect retrieval performances of an image retrieval system.[1,6,8]

D(I ,J) can be denoted as the distance measure between the query image *I* and the image *J* in database, and $f_i(I)$ as the number of pixels in bin i of *I*

6.1 Minkowski-Form Distance

If each dimension of feature vector is independent of each other and is of equal importance, the Minskowski-form distance L_p is a appropriate for calculating the distance between two images. The distance is diefined as [33]

$$D(I,J) = \left(\sum_i |f_i(I) - f_i(J)|^P\right)^{1/p}$$

Where p=1, 2, ∞, D(I, J) is the L1, L2(also called Euclidean distance) and L∞ distance respectively.

6.2 Quadratic Form (QF) Distance

The Minkowski distance treats all bins of the feature histogram entirely independent 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 [6,8,33].

$$D(Q,I) = \sqrt{(H_Q - H_I)^t A (H_Q - H_I)}$$

Where A=[a_{ij}] is a similarity matrix, and a_{ij} denotes the similarity between bin Q and I. H_Q and H_I are vectors that list all the entries in $H_i(Q)$ and $H_i(I)$.

Other similarity measures technique such as Mahalonobis Distance and Kullback-Leibler(KL) Divergence and Feffery-Divergence (JD) are also used for similarity measures.

6.3 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[1]:

$$D(I,J) = \sqrt{(F_I - F_J)^T C^{-1} (F_I - F_J)}$$

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, c_i, is needed.

$$D(I,J) = \sum_{i=1}^{N} (F_I - F_J)^2 / c_i$$

6.4 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:

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

The KL divergence is used in [1,33] as the similarity measure for texture. The *Jeffrey-divergence* (*JD*) is defined by:

$$D(I,J)\,\mathrm{X}\quad_{i}\ f_i(I)\log\frac{f_i(I)}{f_i(J)}\,\Gamma\,f_i(J)\log\frac{f_i(I)}{f_i(J)}$$

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

CHAPTER VII

7. Implementation

This chapter describes the implementation of the algorithm proposed in the previous chapters for the feature extraction and similarity measures. Here in chapter 3, different techniques for the low level feature extraction of the image are discussed. In chapter 4, different texture feature extraction techniques are discussed. This chapter describes the algorithm implementation for extracting the best matched from the database of the system.

7.1 Implementation Tools

Different tools have been used for the implementation processes. For the feature extraction and histogram generation MATLAB is used. The programming has been accomplished by using Visual C++ codes.

7.1.1 MATLAB

MATLAB is the acronym of Mathematical Library. It is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include Math and computation Algorithm development Data acquisition Modeling, simulation, and prototyping Data analysis, exploration, and visualization Scientific and engineering graphics Application development, including graphical user interface building.
The front end design of the application has been achieved by using MATLAB GUIDE.

7.1.2 Data Structures

Data Structures used in this retrieval system are Visual C++ build in Objects.

Vector or Array: It is a collection for storing and manipulating the objects stored in it. In this dissertation, the use of vector is used for the representation of the multi-dimensional matrix for the statistical value storage and manipulation.

String: This is an array of characters. This data structure is used to represent the query image and indexing of the images for the result displaying..

7.2 Colour Feature

7.2.1 Histograms

Global colour histogram is used in extracting the colour features of images. In analyzing the histograms there were a few issues that had to be dealt with. First there was the issue of how the number of bins in a histogram is quantized. By default the number of bins represented in an image's colour histogram using the ***imhist()*** function in MatLab is 256. Meaning that in the calculations of similarity matrix and histogram difference, the processing would be computationally expensive. During the initial implementation, the number of bins is quantized to 20. This means that colours that are distinct yet similar are assigned to the same bin reducing the number of bins from 256 to 20. This obviously decreases the information content of images, but decreases the time in calculating the colour distance between two histograms. On the other hand keeping the number of bins at 256 gives a more accurate result in terms of colour distance.

The second issue was in which colour space to select for the image representation. Should it be **RGB** or **HSV**? This was solved right away when it is found that ***QBIC***'s similarity matrix equation was using the **HSV** colour space in its calculation. There hasn't been any evidence to show which colour space generates the best retrieval results, thus the use of this colour space did not restrict anyway.

7.2.2 Quadratic Distance Metric

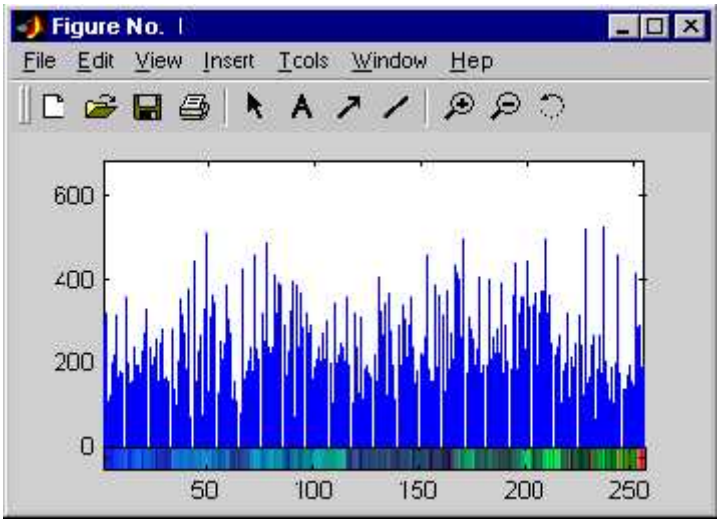
The equation used in deriving the distance between two colour histograms is the quadratic distance metric:

$$D(Q,I)\,\mathrm{X}\sqrt{(H_Q\,\mathrm{Z}\,H_I)^t\,A(H_Q\,\mathrm{Z}\,H_I)}$$

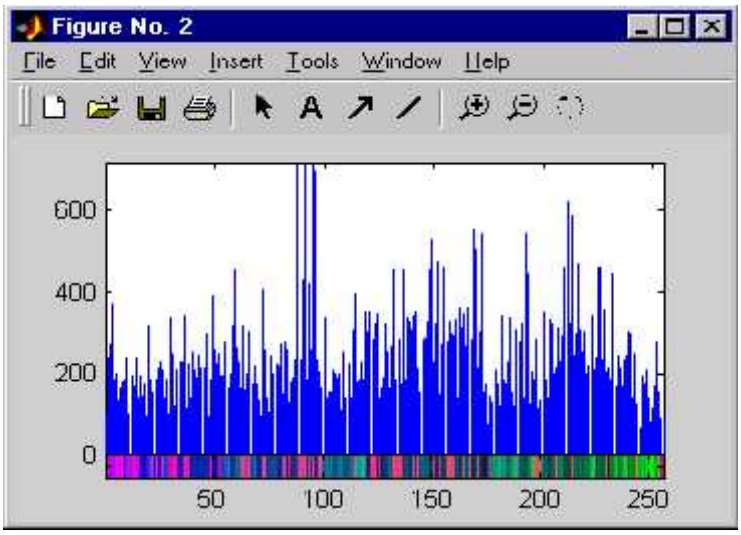
The equation consists of three terms. The derivation of each of these terms will be explained in the following sections. The first term consists of the difference between two colour histograms; or more precisely the difference in the number of pixels in each bin. This term is obviously a vector since it consists of one row. The number of columns in this vector is the number of bins in a histogram. The third term is the transpose of that vector. The middle term is the similarity matrix. The final result **d** represents the colour distance between two images. The closer the distance is to zero the closer the images are in colour similarity. The further the distance from zero the less similar the images are in colour similarity.

7.2.3 Similarity Matrix

As can be seen from the colour histograms of two images ***Q*** and ***I*** in the figure below, the colour patterns observed in the colour bar are totally different. This is further confirmed when one sees the respective colour maps in the following table…



(a) Image Q



(a) Image I

Figure 7.1: *Colour Histograms of two images.*

Colour Map of image Q				***Colour Map of image I***		
0.9608	0.8980	0.7843		0.9922	0.9882	0.9961
0.9373	0.9059	0.8235		0.9569	0.9569	0.9882
0.9098	0.8510	0.7765		0.9725	0.9647	0.9765
0.9255	0.8588	0.8039		0.9176	0.9137	0.9569
0.8627	0.8275	0.7961		0.9098	0.8980	0.9176
0.9098	0.8431	0.7216		0.9569	0.9255	0.9412
0.9137	0.8392	0.6627		0.9020	0.8627	0.8980
0.9059	0.7882	0.6510		0.9020	0.8431	0.8510
0.9451	0.8275	0.6824		0.9098	0.8196	0.8078
0.9569	0.7882	0.5922		0.8549	0.8510	0.8941
0.9137	0.7765	0.5961		0.8235	0.8235	0.8941
0.9412	0.7961	0.5569		0.8471	0.8353	0.8549
.	.	.		.	.	.
.	.	.		.	.	.

Table 7.1: *Colour Maps of two images.*

A simple distance metric involving the subtraction of the number of pixels in the 1st bin of one histogram from the 1st bin of another histogram and so on is not adequate. This metric is referred to as a *Minkowski-Form Distance Metric*, shown below, which only compares the "same bins between colour Histograms"[13].

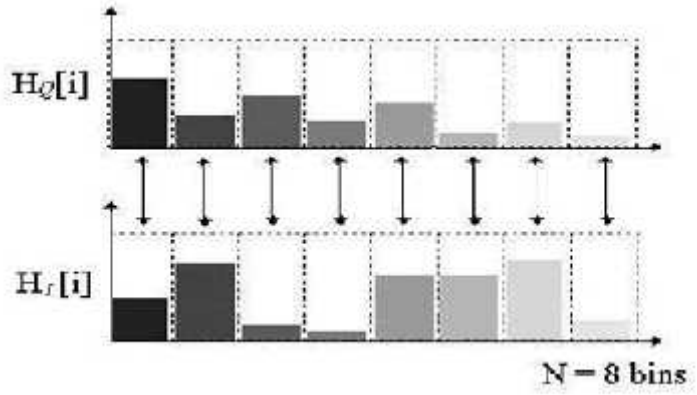


***Figure 7.2** : Minkowski Distance Approach*

This is the main reason for using the quadratic distance metric. More precisely it is the middle term of the equation or similarity matrix ***A*** that helps to overcome the problem of different colour maps. The similarity matrix is obtained through a complex algorithm:

$$a_{q,i} = 1 - \frac{\left[(v_q - v_i)^2 + (s_q \cos h_q - s_i \cos h_i)^2 + (s_q \sin h_q - s_i \sin h_i)^2 \right]^{\frac{1}{2}}}{\sqrt{5}}$$

which basically compares one colour bin of $\boldsymbol{H_Q}$ with all those of $\boldsymbol{H_I}$ to try and find out which colour bin is the most similar, as shown below:

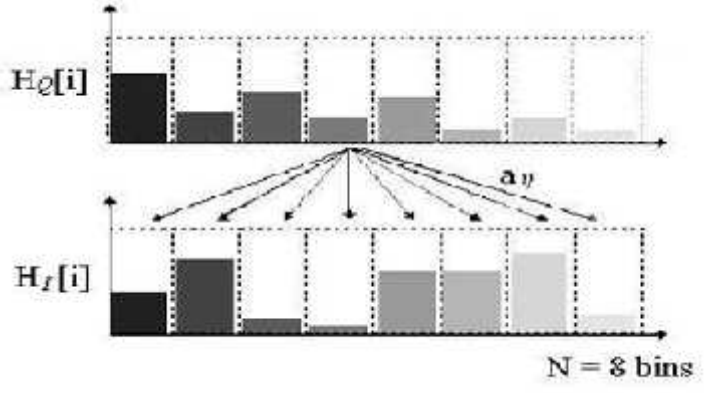


***Figure 7.3**: Quadratic Distance Approach*

This is continued until it compared all the colour bins of H_Q. In doing so an N x N matrix is obtained, N representing the number of bins. What indicates whether the colour patterns of two histograms are similar is the diagonal of the matrix, shown below. If the diagonal entirely consists of ones then the colour patterns are identical. The further the numbers in the diagonal are from one, the less similar the colour patterns are. Thus the problem of comparing totally unrelated bins is solved.

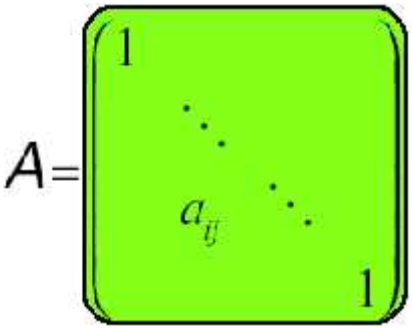


***Figure 7.4:** Similarity Matrix A, with a diagonal of ones*

7.3 Texture Feature

7.3.1 Pyramid-Structured Wavelet Transform

Different texture extraction techniques are found in the literature of the CBIR. It is found that wavelet representation of the texture is quite efficient in the representation of the image. In this work, the pyramid-structured wavelet transform has been used for texture classification. Its name comes from the fact that it recursively decomposes sub signals in the low frequency channels. It is mostly significant for textures with dominant frequency channels. For this reason, it is mostly suitable for signals consisting of components with information concentrated in lower frequency channels. Due to the innate image properties that allows for most information to exist in lower sub-bands, the pyramid-structured wavelet transform is highly sufficient.

Using the pyramid-structured wavelet transform, the texture image is decomposed into four sub images, in low-low(LL), low-high(LH), high-low(HL) and high-high(HH) sub-bands. At this point, the energy level of each sub-band is calculated. This is first level decomposition. Using the low-low sub-band for further decomposition, the fifth level of decomposition has been reached in this method. The reason for this is the basic assumption that the energy of an image is concentrated in the low-low band. For this reason the wavelet function used is the Daubechies wavelet.

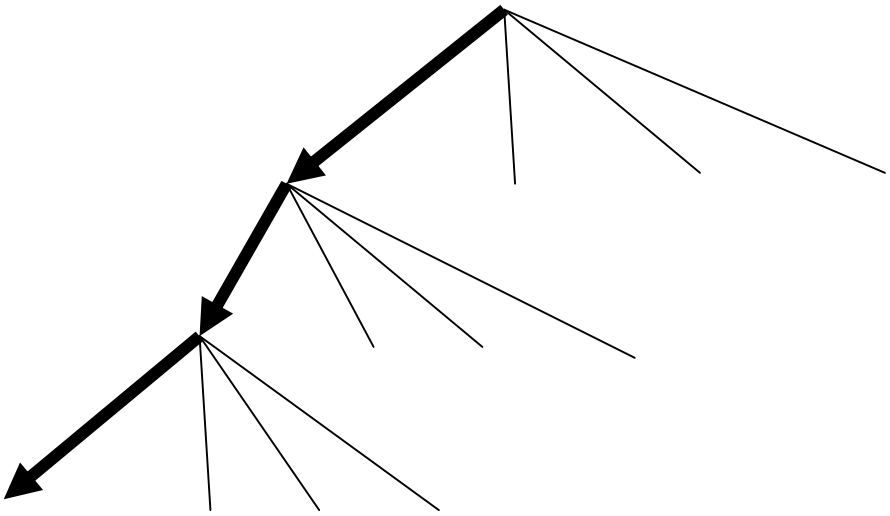


Figure 7.5: *Pyramid-Structured Wavelet Transform.*

7.3.2 Energy Level

Energy Level Algorithm:

1. Decompose the image into *four* sub-images
2. Calculate the energy of all decomposed images at the same scale, using:

$$E \text{ X } \frac{1}{MN} \sum_{i\text{X}1}^{m} \sum_{j\text{X}1}^{n} \left| X\, fi,\, jA \right|$$

where ***M*** and ***N*** are the dimensions of the image, and ***X*** is the intensity of the pixel located at row ***i*** and column ***j*** in the image map.

3. Repeat from step 1 for the low-low sub-band image, until index ***ind*** is equal to 5. Increment ***ind***.

Using the above algorithm, the energy levels of the sub-bands were calculated and further decomposition of the low-low sub-band image. This is repeated five times, to reach fifth level decomposition. These energy level values are stored to be used in the Euclidean distance algorithm.

7.3.3 Euclidean Distance

Euclidean Distance Algorithm:

1. Decompose query image.
2. Get the energies of the first dominant ***k*** channels.
3. For image ***i*** in the database obtain the ***k*** energies.
4. Calculate the Euclidean distance between the two sets of energies, using:

$$D_i \; \mathrm{X} \sum_{k\,\mathrm{X}1}^{k} f x_k \; \mathrm{Z} \; y_{i,k} \text{Å}^2$$

5. Increment ***i***. Repeat from step 3.

Using the above algorithm, the query image is searched for in the image database. The Euclidean distance is calculated between the query image and every image in the database. This process is repeated until all the images in the database have been compared with the query image. Upon completion of the Euclidean distance algorithm, an array of Euclidean distances is obtained, which is then sorted. The five topmost images are then displayed as a result of the texture search.

7.4 Graphical User Interface (GUI)

The Graphical User Interface was constructed using ***MatLab GUIDE*** or ***Graphical User Interface Design Environment***. Using the layout tools provided by ***GUIDE***, the graphical user

interface (**aMain.fig**) is designed for the CBIR application

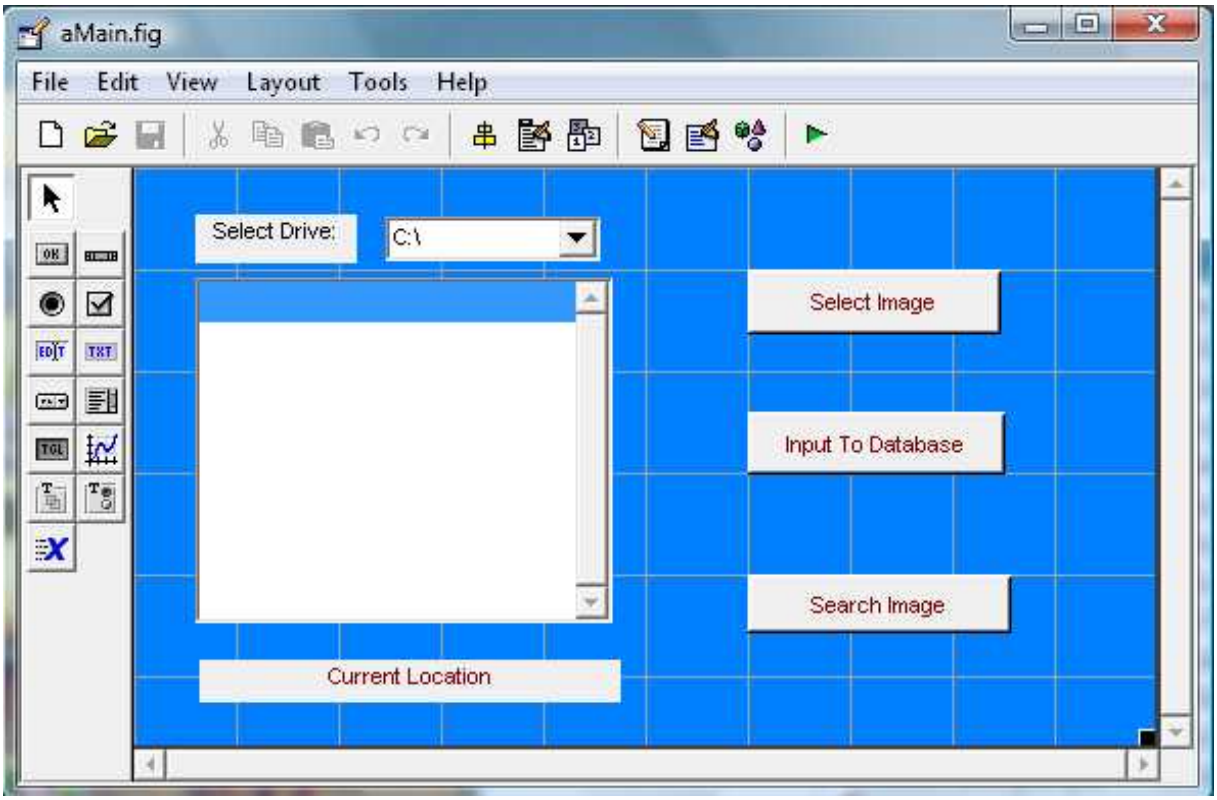


***Figure 7.6**: GUI Design*

In addition to the above outlined design, a simple menu structure was also designed, using the ***Menu Editor***, as shown below:

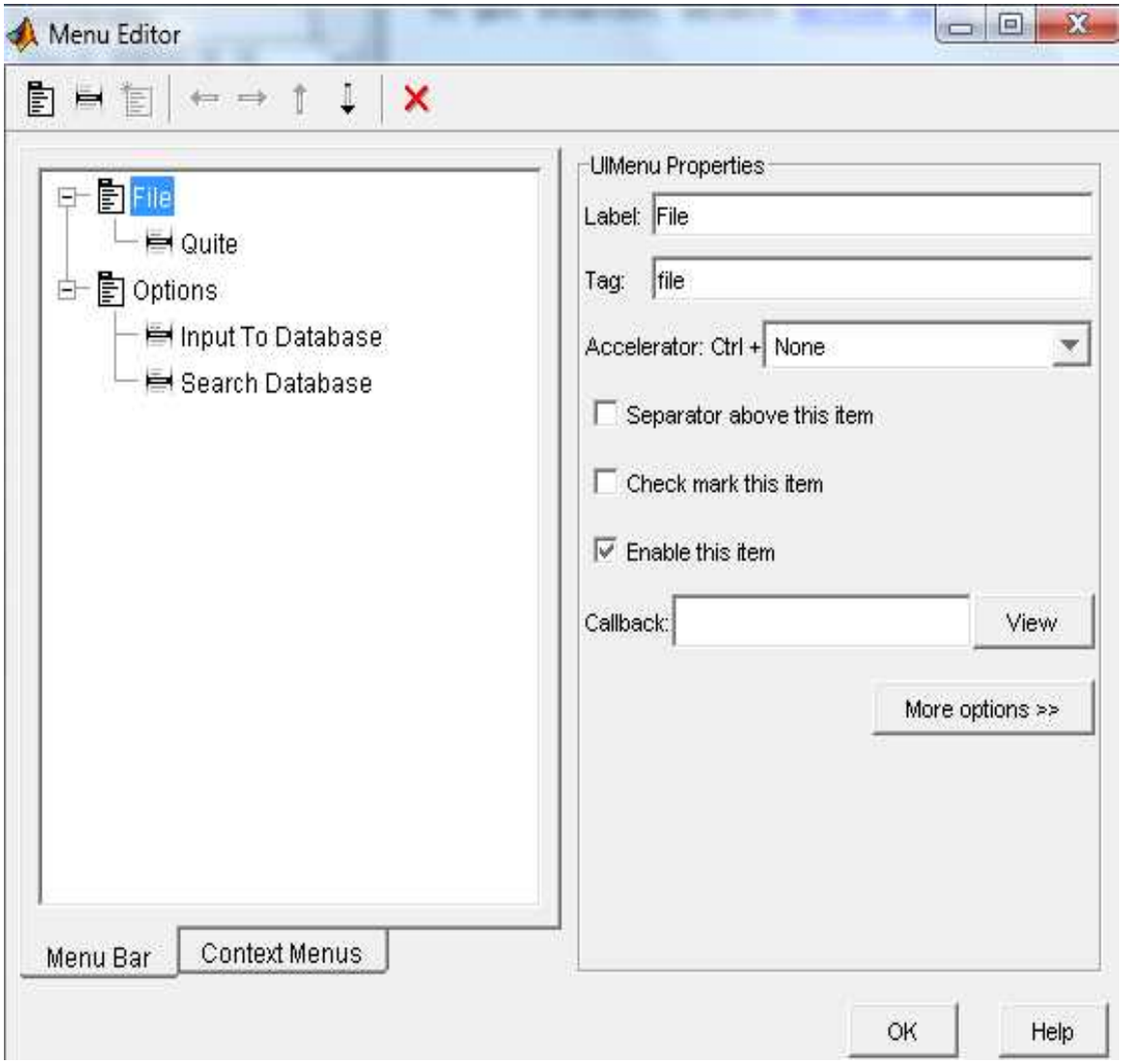


Figure7.7: *Menu Editor specification for the menu…*

The above design yields the following application window on run time:

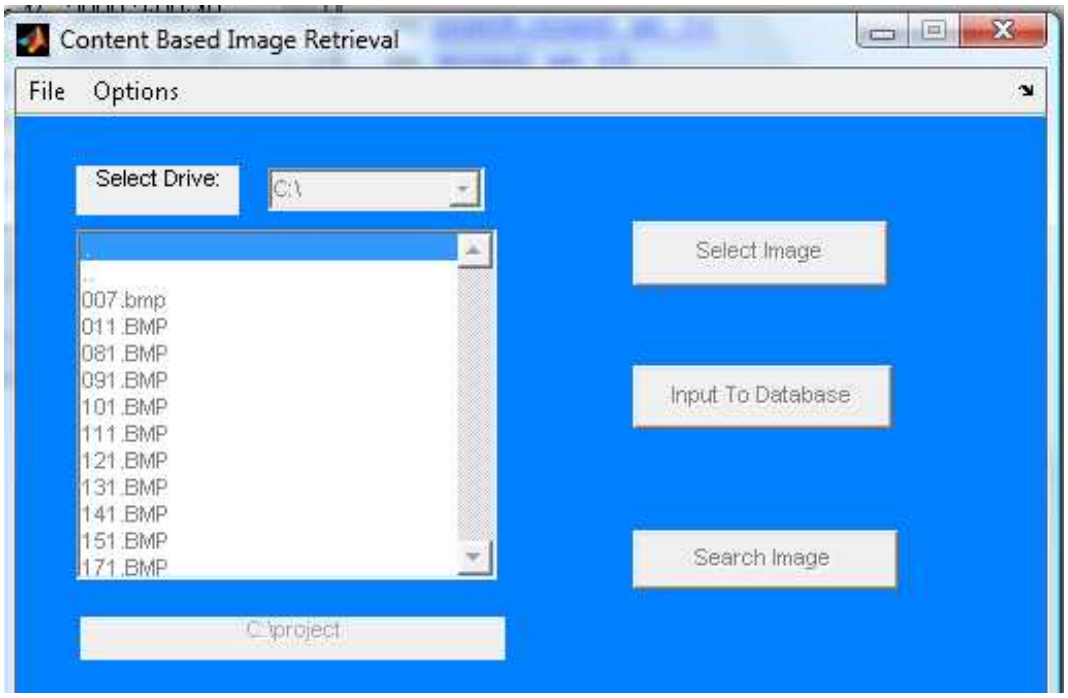


Figure7.8: *Application window at runtime*

The handlers for clicking on the buttons are coded using MatLab code to perform the necessary operations.

7.5 Database

The image database that are used in this thesis contains sixty 8-bit uncompressed it maps ***BMP***s that have been randomly selected from the World Wide Web. The following figure depicts a sample of images in the database:

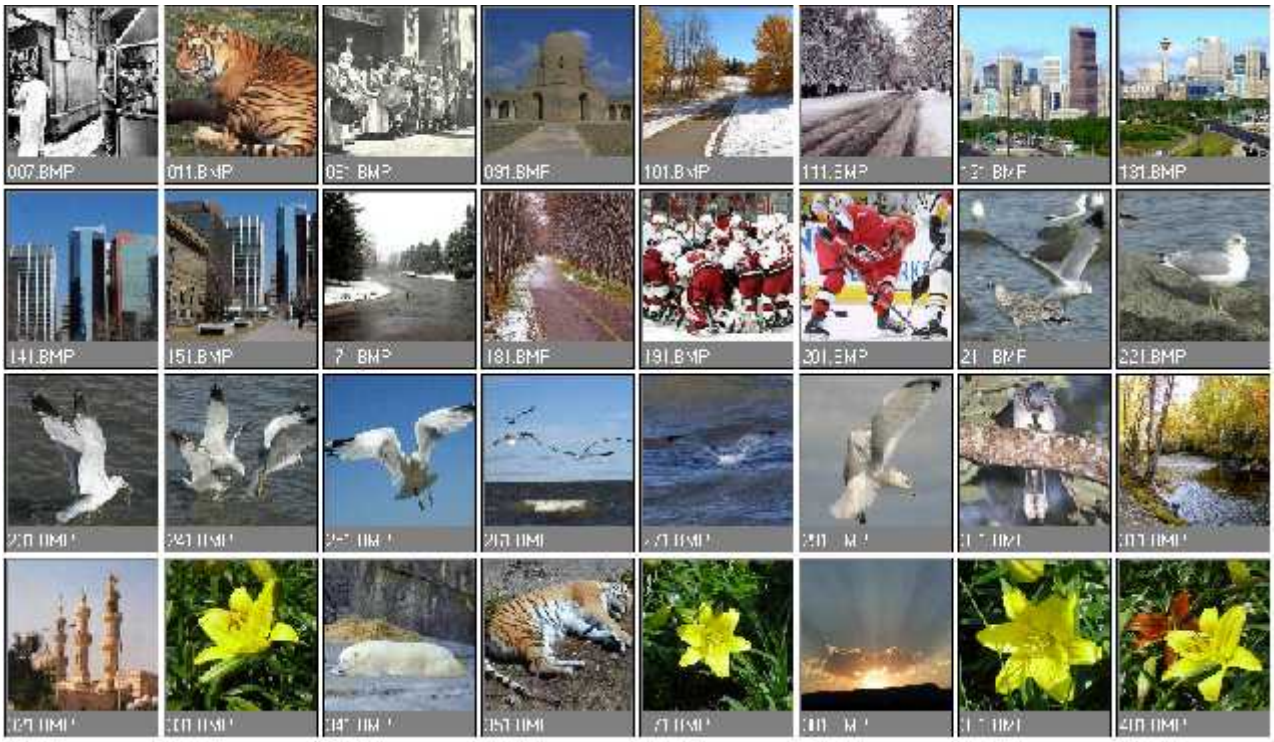


Figure 7.9: *Image Database*

CHAPTER VIII

8. Testing and Analysis

To demonstrate the result of application, a query image was passed through the system:

- The application was started by typing ***aMain*** and pressing return in the MatLab Command Window. The application window started.
- In the application window, The Search databse option was selected from the *Options* menu. This enabled the browsing window, to browse to a *BMP* file.
- Upon highlighting a *BMP* file, the select button became enabled. **Note:** Only 8-bit uncompressed *BMP*s are suitable for this application. In this example, the image named with 371.bmp was .
- The highlighted *BMP* is then selected by pressing the *Select* button.
- Next, pressing the *Search* button started the search.

Figure 8.1: *The query image: 371.bmp…*

8.1 Colour Extraction & Matching

Using the *colour feature extraction algorithm* described above, where the histograms of the query image and the images in the database are compared using the *Quadratic Distance Metric*, the following top 10 results are displayed.

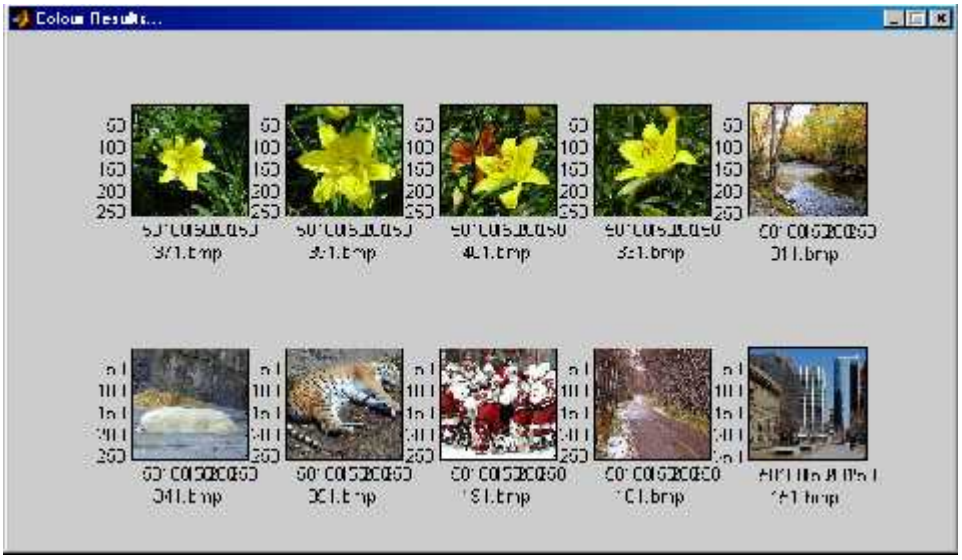


Figure 8.2 : *Colour Results for the searching for 371.bmp*

The above results are sorted according to the quadratic distance… These are shown below:

File Name	***Colour Distance***
371.bmp	0
391.bmp	3.3804
401.bmp	4.3435
331.bmp	5.0800
311.bmp	5.9940
341.bmp	6.6100
351.bmp	6.9638
191.bmp	7.0813
181.bmp	7.1060
151.bmp	7.1958

Table 8.1: *Colour distance between query and results*

8.2 Texture Extraction & Matching

Using the *texture feature extraction algorithm* described above, where the energies of the query image and the *colour result* images' sub-bands are compared using the *Euclidean Distance Metric*, the following top 4 results are obtained

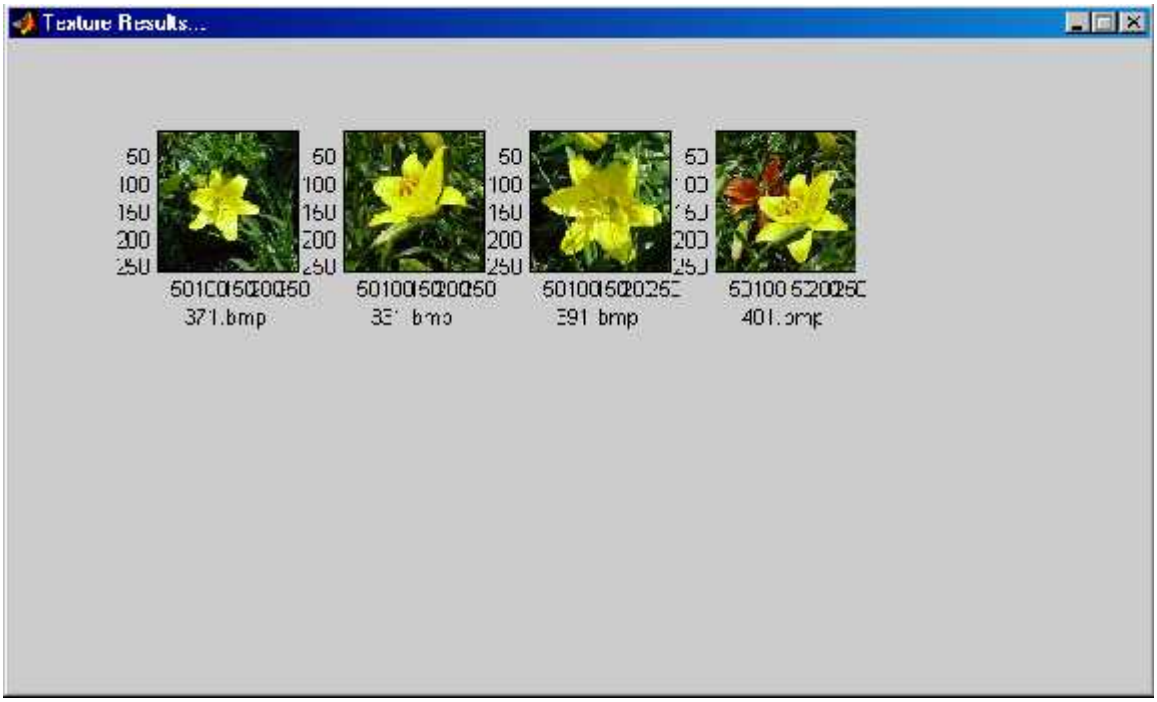


Figure 8.3: *Texture Results for the searching for 371.bmp*

The above results are sorted according to the Euclidean distance… These are shown below:

File Name	***Euclidean Distance***
371.bmp	0
331.bmp	1.1449
391.bmp	2.4609
401.bmp	2.6926

Table 8.2: *Euclidean distance between query and results…*

By observing the images in database, It can be found that the above results represent the closest matches to the query image chosen.

1.5 Performance Evaluation

To evaluate the performance of retrieval system, two measurements, namely, *recall* and *precision* [1], are borrowed from traditional information retrieval. For a query q, the data set of images in the database that are relevant to the query q is denoted as $R(q)$, and the retrieval result of the query q is denoted as $Q(q)$. The precision of the retrieval is defined as the fraction of the retrieved images that are indeed relevant for the query:

$$precision \; \mathrm{X} \frac{|Q(q) IR(q)|}{|Q(q)|}$$

The recall is the fraction of relevant images that is returned by the query:

$$recall \; \mathrm{X} \frac{|Q(q) IR(q)|}{|R(q)|}$$

Usually, a tradeoff must be made between these two measures since improving one will sacrifice the other. In typical retrieval systems, recall tends to increase as the number of retrieved items increases; while at the same time the precision is likely to decrease. In addition, selecting a relevant data set $R(q)$ is much less stable due to various interpretations of the images. Further, when the number of relevant images is greater than the number of the retrieved images, recall is meaningless. As a result, precision and recall are only rough descriptions of the performance of the retrieval system.

CHAPTER IX

9. Conclusions and Further Recommendation

9.1 Summary

The dramatic rise in the sizes of images databases has stirred the development of effective and efficient retrieval systems. The development of these systems started with retrieving images using textual connotations but later introduced image retrieval based on content. This came to be known as CBIR or Content Based Image Retrieval. Systems using CBIR retrieve images based on visual features such as colour, texture and shape, as opposed to depending on image descriptions or textual indexing. In this dissertation, various modes of representation and retrieval of the image properties of colour, texture and shape are researched as the core of feature extraction techniques.

The application performs a simple colour-based search in an image database for an input query image, using colour histograms. It then compares the colour histograms of different images using the *Quadratic Distance Equation*. Further enhancing the search, the application performs a texture-based search in the colour results, using wavelet decomposition and energy level calculation. It then compares the texture features obtained using the *Euclidean Distance Equation*. A more detailed step would further enhance these texture results, using a shape-based search.

CBIR is still a developing science. As image compression, digital image processing, and image feature extraction techniques become more developed, CBIR maintains a steady pace of development in the research field. Furthermore, the development of powerful processing power and faster and cheaper memories contribute heavily to CBIR development. This development promises an immense range of future applications using CBIR.

9.2 Further Recommendation

In CBIR System, the image can be retrieved on the basis of the color, texture and shape features. The methods of representation of the image are done in this dissertation basically on the basic global feature description. Image segmentation technique can also be used to represent images' local feature description so the implemented to get finer image representation and comparison can be achieved on the basis of local images description. This technique can enhance the retrieval system by a margin.

Furthermore, human perception of image similarity is subjective, semantic, and task-dependent. Although Content-based methods provide promising directions for image retrieval, generally, the retrieval results based on the similarities of pure visual features are not necessarily perceptually and semantically meaningful. In addition, each type of visual feature tends to capture only one aspect of image property and it is usually hard for a user to specify clearly how different aspects are combined. To address these problems, interactive *relevance feedback*, a technique in traditional text-based information retrieval systems, can be used. With relevance feedback, it is possible to establish the link between high-level concepts and low-level features to retrieve the images.

References:

[1] Long F., Zhang H., Feng D. D., *Fundamentals of Content-based Image Retrieval.* http://www.research.microsoft.com/asia/dload_files/group/mcomputing/2003P

[2] Lew M. S., Sebe N. , Djeraba C. , Jain R., *Content-Based Multimedia Information Retrieval: State of the Art and Challenges*, ACM Transactions on Multimedia computing, Communications and Applications, Vol.2 No.1, February 2006, pp 1–19.

[3] Veltkamp R. C., Tanase M., *Content-Based Image Retrieval Systems: A Survey Technical Report UU-CS-2000-34*, October 2000.

[4] Zhang D. S. and Lu G., *Review of Shape Representation and Description Techniques. Pattern Recognition*, ACM 2004, pp1-19.

[5] Koskela M., Laaksonen J., Oja E, *Entropy-based Measures for Clustering and SOMTtopology preservation applied to Content-Based Image Indexing and Retrieval*, Proceedings of the 17th International Conference on Pattern Recognition ICPR, 2004.

[6] McDonald S., Tait J., *Search Strategies in Content-Based Image Retrieval*, ACM, 2003, pp1-58, 113-646.

[7] Huang J., Kumar S. R., Mitra M., *Combining Supervised Learning with Color Correlograms for Content-Based Image Retrieval*, ACM Multimedia 97 Seattle Washington USA.

[8] Abbadeni N., *Content Representation and Similarity Matching for Texture-based Image Retrieval, MIR'03*, November 7, 2003, Berkeley, California, USA, ACM 2003.

[9] Datta R., Joshi D., Li J., and Wang J. Z., *Image retrieval: Ideas, influences, and trends of the new age*. ACM Computing Survey. Vol 40 No. 2, April 2008.

[10] Su Z., Li S., Zhang H., *Extraction of Feature subspaces for Content-based Retrieval Using Relevance Feedback*, ACM , 2001.

[11] Tzelepi S. K., Koukopoulos D. K., Pangalos G., *A flexible content and context-based access control model for multimedia medical image database systems,* ACM 2001, pp 1-58,113-393.

[12] Pass G., Zabih R., Miller J., *Comparing Images Using Color Coherence Vectors,* Cornell University
http://www.cs.cornell.edu/home/rdz/ccv.html

[13] Kang K., Yoon Y., Choi J., *Additive Texture Information Extraction Using Color Coherence Vector,* 7th WSEAS International Conference on Multimedia Systems & Signal Processing, Hangzhou, China, April 15-17, 2007.

[14] Brandt S., Laaksonen J., Oja E., *Statistical Shape Features in Content-based Image Retrieval , Proceedings of ICPR2000*, Barcelona, Spain, September 2000.

[15] Yu C. T., Aslandogan Y. A., *Multiple evidence combination in image retrieval: Diogenes searches for people on the web.* In Proceedings of 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM 2000, pp 88–95.

[16] Zhang D., Lu G., *Generic Fourier Descriptor for Shape-based Image Retrieval*
Gippsland School of Computing and Info Tech , Monash University, Churchill, Victoria 3842

[17] Wang J. Z., Boujemma N., Delbimbo A., Geman D., Auptmann A., And Tesic J., *Diversity in Multimedia Information Retrieval Research*. In Proceedings of the ACM SIGMM

International Workshop on Multimedia Information Retrieval (MIR) at the International Conference on Multimedia 2006.

[18] Vasconcelos N., *On the efficient evaluation of probabilistic similarity functions for image retrieval*.IEEE Trnas. Inf. Theory, Vol 50 No 7, IEEE 2004, pp1482–1496.

[19] El-Kwae E. A., Kabuka M. R., *Efficient Content-based Indexing of Large Image Databases,* ACM Transactions on Information Systems, Vol. 18 No. 2, April 2000, pp 171–210.

[20] Foschi P. G., Kolippakkam D., Liu H., Mandvikar A., *Feature Extraction for Image Mining,* 8th International Workshop on Multimedia, 2000.

[21] Hearn D., Baker M. P., *Computer Graphics C Version, 2nd Edition*, Pearson Education, 1997.

[22] Pass G., Zabih R., *Comparing images using joint histograms*, Multimedia Systems, Vol.7, 1999, pp 234-240.

[23] Huang J., Ravi Kumar S., Mitra M., *Combining Supervised Learning with Color Correlograms for Content-Based Image Retrieval*, ACM multimedia Seattle Washington USA, 1997.

[24] Zhang H. et al, *Image retrieval based on color features: an evaluation study*. Proceedings of SPIE. Vol. 26 No. 06, 2007, pp 212-220.

[25] Keen N., Fisher B., *Color Moments, Ferbruary 09, 2005.* http://homepages.inf.edu.ac.uk/rbf/CVonline/LOCAL_COPIES/AV045/KEEN.

[26] Weber R., Mlivoncic M., *Efficient Region-baded Image Retrieval,* ACM, 2003.

[27] Howarth P., Yavlinsky A., Heesch D., Ruger S., *Visual Feature for Content-based Medical Image Retrieval,* Multimedia Information Retrieval Team, Imapaerial College London, UK.

[28] Haar Wavelet. http://amath.colorado.edu/courses/4720/2000Spr/Labs/Haar/haar.html

[29] Daubechies wavelet. http://amath.colorado.edu/courses/4720/2000Spr/Labs/DB/db.html

[30] Yu H., Li M., Zhang H. and Feng J., *Color texture moment for content-based image retrieval*, Proc. IEEE Intl Conf. on Image Processing, September, 2002.

[31] Keysers D., Dahmen J., Ney H., Wein B. B., Lehmann T. M. *: A statistical framework for model-based image retrieval in medical applications*, Journal of Electronic Imaging, Vol 12 No 1, 2003.

[32] Lehmann T. M., Schubert H., Keysers D., Kohnen M., Wein B. B., *The IRMA code for unique classification of medical images.* Proceedings SPIE 2003; 5033: in press in this issue.

[33] Seidl T., Kriegel H., *Efficient User-Adaptable Similarity Search in Large Multimedia Databases,* Proceedings of the 23rd VWB Conference Athens, Greece, 1997.

[34] Zheng B., Macclean D. C., And Lu X., *Identifying biological concepts from a protein-related corpus with a probabilistic topic model*. *BMC Bioinformatics. 7*, 58, 2006.

[35] Zhang R., and Zhang Z., *Hidden semantic concept discovery in region based image retrieval.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[36] Chun J., Stockman G., *Subband Image Segmentation Using VQ for Content-Based Image Retrieval, MM'OI,* Ottawa, Canada., Sept. 30-Oct. 5, ACM 2001 ppl-581.

[37] Kherfi M. L., Ziou D., Bernardi A., *Image Retrieval from the World Wide Web: Issue, Techniques, and System, ACM Computing Surveys*, Vol. 36, No 1, March 2004, pp 35-67.

[38] Lafore R., *Object Oriented Programming in Microsoft C++*, The Waite Group, 1995.

[39] *MATLAB Documentation*, Version 7.1.0.246 Service Pack 3.

ANNEX- Matlab Code

aMain.m

```
function varargout = Main(varargin)
```

```
% ------------------------------------------------------------
%   MAIN Application M-file for Main.fig
%   MAIN, by itself, creates a new MAIN or raises the existing
%   singleton*.
%
%   H = MAIN returns the handle to a new MAIN or the handle to
%   the existing singleton*.
%
%   MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in MAIN.M with the given input arguments.
%
%   MAIN('Property','Value',...) creates a new MAIN or raises the
%   existing singleton*.  Starting from the left, property value pairs are
%   applied to the GUI before Main_OpeningFunction gets called.  An
%   unrecognized property name or invalid value makes property application
%   stop.  All inputs are passed to Main_OpeningFcn via varargin.
%
%   *See GUI Options - GUI allows only one instance to run (singleton).
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Main

% Last Modified by GUIDE v2.5 21-Jun-2008 03:11:52
% ------------------------------------------------------------
% ------------------------------------------------------------
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',     gui_Singleton, ...
```

```
                'gui_OpeningFcn',    @Main_OpeningFcn, ...
                'gui_OutputFcn',     @Main_OutputFcn, ...
                'gui_LayoutFcn',     [], ...
                'gui_Callback',      []);

if nargin == 0   % LAUNCH GUI
        initial_dir = pwd;

% Open FIG-file
fig = openfig(mfilename,'reuse');       % Generate a structure of handles to pass to callbacks, and
store it.
handles = guihandles(fig);
guidata(fig, handles);
%disp('populate1!!');
% Populate the listbox
load_listbox(initial_dir,handles)
% Return figure handle as first output argument
   if nargout > 0
        varargout{1} = fig;
   end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
 try
   [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
 catch
   disp(lasterr);
 end
end

% End initialization code - DO NOT EDIT
% --------------------------------------------------------------
```

```
% --------------------------------------------------------------
% Executes just before Main is made visible.
% --------------------------------------------------------------
function Main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Main (see VARARGIN)

% Choose default command line output for Main...
handles.output = hObject;

% Initialize the options...
handles.option = 'input';

% Update handles structure...
guidata(hObject, handles);

% --------------------------------------------------------------


% --------------------------------------------------------------
% Outputs from this function are returned to the command line.
% --------------------------------------------------------------
function varargout = Main_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% ------------------------------------------------------------


% ------------------------------------------------------------
% Callback for list box - open .fig with guide, otherwise use open
% ------------------------------------------------------------
function varargout = listbox1_Callback(h, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1
mouse_event = get(handles.figure1,'SelectionType');
index_selected = get(handles.listbox1,'Value');
file_list = get(handles.listbox1,'String');
filename = file_list{index_selected};
if strcmp(mouse_event,'normal')
   if ~handles.is_dir(handles.sorted_index(index_selected))
      [newpath, name, ext, ver] = fileparts(filename);
      switch ext
         case '.BMP'
            set(handles.selectButton, 'Enable', 'On');
         case '.bmp'
            set(handles.selectButton, 'Enable', 'On');
         otherwise
            set(handles.selectButton, 'Enable', 'Off');
      end
```

```
   end
end

if strcmp(mouse_event,'open')
        if  handles.is_dir(handles.sorted_index(index_selected))
                cd (filename)
                load_listbox(pwd,handles)
   end
end

% --------------------------------------------------------------------


% --------------------------------------------------------------------
% Read the current directory and sort the names
% --------------------------------------------------------------------
function load_listbox(dir_path,handles)
cd (dir_path)
dir_struct = dir(dir_path);
[sorted_names,sorted_index] = sortrows({dir_struct.name}');
handles.file_names = sorted_names;
handles.is_dir = [dir_struct.isdir];
handles.sorted_index = [sorted_index];
guidata(handles.figure1,handles)
set(handles.listbox1,'String',handles.file_names,...
        'Value',1)
set(handles.text1,'String',pwd)

% --------------------------------------------------------------------
```

```
% --------------------------------------------------------------------
% Executes during object creation, after setting all properties.
% --------------------------------------------------------------------
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --------------------------------------------------------------------



% --------------------------------------------------------------------
% Executes during object creation, after setting all properties.
% --------------------------------------------------------------------
function popupmenu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --------------------------------------------------------------




% --------------------------------------------------------------
% Executes on selection change in popupmenu.
% --------------------------------------------------------------
function popupmenu_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu

val = get(hObject,'Value');
str = get(hObject, 'String');
cd(str{val});

load_listbox(pwd, handles)

% --------------------------------------------------------------
```

```
% --------------------------------------------------------------
% Executes on selection of 'Quite' from the menubar.
% --------------------------------------------------------------
function quite_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

selection = questdlg('Are you sure you want to quite?',...
                  ['Close ' get(handles.figure1,'Name') '...'],...
                  'Yes','No','Yes');
if strcmp(selection,'No')
   return;
end

delete(handles.figure1)

% --------------------------------------------------------------



% --------------------------------------------------------------
% Executes on selection of 'Input To Database' from the menubar.
% --------------------------------------------------------------
function inputDatabase_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Some code to input the selected image to the database...

set(handles.input, 'Checked', 'On');
set(handles.search, 'Checked', 'Off');

set(handles.listbox1, 'Enable', 'On');
set(handles.text1, 'Enable', 'On');
set(handles.popupmenu, 'Enable', 'On');

handles.option = 'input';       % This means that the option is to "input to database"...

guidata(hObject, handles)

% --------------------------------------------------------------




% --------------------------------------------------------------
% Executes on selection of 'Search Database' from the menubar.
% --------------------------------------------------------------
function searchDatabase_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Some code to search the database for the selected image...
set(handles.input, 'Checked', 'Off');
set(handles.search, 'Checked', 'On');
```

```
set(handles.listbox1, 'Enable', 'On');
set(handles.text1, 'Enable', 'On');
set(handles.popupmenu, 'Enable', 'On');

handles.option = 'search';      % This means that the option is to "search database"...
guidata(hObject, handles)
% --------------------------------------------------------------------


% --------------------------------------------------------------------
% Executes on button press in selectbutton.
% --------------------------------------------------------------------
function selectButton_Callback(hObject, eventdata, handles)
% hObject    handle to selectButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
index_selected = get(handles.listbox1,'Value');
file_list = get(handles.listbox1,'String');
filename = file_list{index_selected};
[newpath,name,ext,ver] = fileparts(filename);
handles.filename = strcat(name,ext);
[handles.queryx, handles.querymap] = imread(filename);    %read the image file.
cd('C:\Project');
figure
imshow(handles.queryx, handles.querymap); %This displays the image.
% Obtain HSV format of the image...
handles.queryhsv = rgb2hsv(handles.querymap);
guidata(hObject,handles)
set(handles.selectButton, 'Enable', 'Off');
set(handles.listbox1, 'Enable', 'Off');
```

```
set(handles.text1, 'Enable', 'Off');
set(handles.popupmenu, 'Enable', 'Off');
set(handles.input, 'Enable', 'Off');
set(handles.search, 'Enable', 'Off');
% handles.option
switch handles.option
case 'input'
   set(handles.inputButton, 'Enable', 'On');
case 'search'
   set(handles.searchButton, 'Enable', 'On');
end
% --------------------------------------------------------------



% --------------------------------------------------------------
% Executes on button press in inputButton.
% --------------------------------------------------------------
function inputButton_Callback(hObject, eventdata, handles)
% hObject    handle to transform1button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.inputButton, 'Enable', 'Off');
% Open database txt file... for reading...
fid = fopen('database.txt');
exists = 0;
while 1
   tline = fgetl(fid);
   if ~ischar(tline), break, end       % Meaning: End of File...
   if (strcmp(tline, handles.filename))
      exists = 1;
```

```
        break;
    end
end
fclose(fid);

if ~exists
    fid = fopen('database.txt', 'a');
    fprintf(fid,'%s\r',handles.filename);
    fclose(fid);
end

guidata(hObject, handles)
msgbox('Database updated with file name...', 'Success...');
set(handles.input, 'Checked', 'Off');
set(handles.search, 'Checked', 'Off');
set(handles.input, 'Enable', 'On');
set(handles.search, 'Enable', 'On');
% --------------------------------------------------------------------



% --------------------------------------------------------------------
% Executes on button press in searchButton.
% --------------------------------------------------------------------
function searchButton_Callback(hObject, eventdata, handles)
% hObject    handle to transform2button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.searchButton, 'Enable', 'Off');
% Colour search...
% Open database txt file... for reading...
```

```
fid = fopen('database.txt');
resultValues = [];      % Results matrix...
resultNames = {};
i = 1;                  % Indices...
j = 1;
while 1
   imagename = fgetl(fid);
   if ~ischar(imagename), break, end       % Meaning: End of File...
   [X, RGBmap] = imread(imagename);
   HSVmap = rgb2hsv(RGBmap);
   D = quadratic(handles.queryx, handles.querymap, X, HSVmap);
   resultValues(i) = D;
   resultNames(j) = {imagename};
   i = i + 1;
   j = j + 1;
end
fclose(fid);

% Sorting colour results...
[sortedValues, index] = sort(resultValues);     % Sorted results... the vector index
                                    % is used to find the resulting files.
fid = fopen('colourResults.txt', 'w+');         % Create a file, over-write old ones.
for i = 1:10        % Store top 10 matches...
   tempstr = char(resultNames(index(i)));
   fprintf(fid, '%s\r', tempstr);
   disp(resultNames(index(i)));
   disp(sortedValues(i));
   disp('  ');
end
fclose(fid);
```

```
%return;
disp('Colour part done...');
disp('Colour results saved...');
disp('');
displayResults('colourResults.txt', 'Colour Results...');
disp('Texture part starting...');
% Texture search...
queryEnergies = obtainEnergies(handles.queryx, 6);          % Obtain top 6 energies of the image.
% Open colourResults txt file... for reading...
fid = fopen('colourResults.txt');
fresultValues = [];      % Results matrix...
fresultNames = {};
i = 1;                % Indices...
j = 1;
while 1
   imagename = fgetl(fid);
   if ~ischar(imagename), break, end       % Meaning: End of File...
   [X, RGBmap] = imread(imagename);
   imageEnergies = obtainEnergies(X, 6);
   E = euclideanDistance(queryEnergies, imageEnergies);
   fresultValues(i) = E;
   fresultNames(j) = {imagename};
   i = i + 1;
   j = j + 1;
end
fclose(fid);
disp('Texture results obtained...');
% Sorting final results...
[sortedValues, index] = sort(fresultValues);     % Sorted results... the vector index
                                                 % is used to find the resulting files.
```

```
fid = fopen('textureResults.txt', 'w+');        % Create a file, over-write old ones.
for i = 1:4       % Store top 5 matches...
   imagename = char(fresultNames(index(i)));
   fprintf(fid, '%s\r', imagename);
   disp(imagename);
   disp(sortedValues(i));
   disp('  ');
end
fclose(fid);
disp('Texture results saved...');
displayResults('textureResults.txt', 'Texture Results...');
guidata(hObject,handles)
set(handles.input, 'Checked', 'Off');
set(handles.search, 'Checked', 'Off');
set(handles.input, 'Enable', 'On');
set(handles.search, 'Enable', 'On');
% --------------------------------------------------------
```

decompose.m

```
% Works to decompose the passed image matrix...
% --------------------------------------------------------
% Executes on being called, with input image matrix.
% --------------------------------------------------------
function [Tl, Tr, Bl, Br] = decompose(imMatrix)
[A,B,C,D] = dwt2(imMatrix,'db1');
Tl = wcodemat(A);      % Top left...
Tr = wcodemat(B);      % Top right...
Bl = wcodemat(C);      % Bottom left...
```

```
Br = wcodemat(D);       % Bottom right...
% Display the image decomposition... [For testing purposes]
%figure
%colormap(gray)
%imagesc([Tl, Tr; Bl, Br]);
% ------------------------------------------------------------
```

quadratic.m

```
% Works to obtain the Quadratic Distance between two Colour images
% ------------------------------------------------------------
% Executes on being called, with inputs:
%   X1 - number of pixels of 1st image
%   X2 - number of pixels of 2nd image
%   map1 - HSV colour map of 1st image
%   map2 - HSV colour map of 2nd image
% ------------------------------------------------------------
function value = quadratic(X1, map1, X2, map2)

% Obtain the histograms of the two images...
[count1, y1] = imhist(X1, map1);
[count2, y2] = imhist(X2, map2);

% Obtain the difference between the pixel counts...
q = count1 - count2;
s = abs(q);
% Obtain the similarity matrix...
A = similarityMatrix(map1, map2);
% Obtain the quadratic distance...
d = s.'*A*s;
d = d^1/2;
```

```
d = d / 1e8;
% Return the distance metric.
value = d;
% ------------------------------------------------------------
```

similarityMatrix.m

```
% Works to obtain the Similarity Matrix between two HSV color
% histograms. This is to be used in the Histogram Quadratic
% Distance equation.
% ------------------------------------------------------------
% Executes on being called, with input matrices I and J.
% ------------------------------------------------------------
function value = similarityMatrix(I, J)

% Obtain the Matrix elements... r - rows, c - columns. The
% general assumption is that these dimentions are the same
% for both matrices.
[r, c] = size(I);
A = [];
for i = 1:r
   for j = 1:r
      % (sj * sin hj - si * sin hi)^2
      M1 = (I(i, 2) * sin(I(i, 1)) - J(j, 2) * sin(J(j, 1)))^2;
      % (sj * cos hj - si * cos hi)^2
      M2 = (I(i, 2) * cos(I(i, 1)) - J(j, 2) * cos(J(j, 1)))^2;
      % (vj - vi)^2
      M3 = (I(i, 3) - J(j, 3))^2;
      M0 = sqrt(M1 + M2 + M3);
       %A(i, j) = 1 - 1/sqrt(5) * M0;
```

```
        A(i, j) = 1 - (M0/sqrt(5));
    end
end
 %Obtain Similarity Matrix...
value = A;
% ------------------------------------------------------------
```

obtainEnergies.m

```
% Works to obtain the first 'n' energies of the passed grayscale image...
% ------------------------------------------------------------
% Executes on being called, with input matrix & constant 'n'.
% ------------------------------------------------------------
function value = obtainEnergies(iMatrix, n)

dm = iMatrix;       % The matrix to be decomposed...
energies = [];
i = 1;
for j = 1:5
    [tl, tr, bl, br] = decompose(dm);
    energies(i) = energyLevel(tl);
    energies(i+1) = energyLevel(tr);
    energies(i+2) = energyLevel(bl);
    energies(i+3) = energyLevel(br);
    i = i + 4;
    dm = tl;
end
%Obtain array of energies...
sorted = -sort(-energies);      % Sorted in descending order...
value = sorted(1:n);
% ------------------------------------------------------------
```

energyLevel.m

```
% Works to obtain the energy level of the passed matrix...
% ------------------------------------------------------------
% Executes on being called, with input matrix.
% ------------------------------------------------------------
function value = energyLevel(aMatrix)

% Obtain the Matrix elements... r - rows, c - columns.
[r, c] = size(aMatrix);

%Obtain energyLevel...
value = sum(sum(abs(aMatrix)))/(r*c);
% ------------------------------------------------------------
```

euclideanDistance.m

```
% Works to obtain the Euclidean Distance of the passed vector...
% ------------------------------------------------------------
% Executes on being called, with input vectors X and Y.
% ------------------------------------------------------------
function value = euclideanDistance(X, Y)

[r, c] = size(X);       % The length of the vector...
e = [];
% Euclidean Distance = sqrt [ (x1-y1)^2 + (x2-y2)^2 + (x3-y3)^2 ...]
for i = 1:c
   e(i) = (X(i)-Y(i))^2;
end
```

```
Euclid = sqrt(sum(e));
%Obtain energyLevel...
value = Euclid;

% ------------------------------------------------------------
```

displayResults.m

```
% Works to display the images named in a text file passed to it...
% ------------------------------------------------------------
% Executes on being called, with inputs:
%       filename -  the name of the text file that has the
%                   list of images
%       header -    the figure header name
% ------------------------------------------------------------
function displayResults(filename, header)

figure('Position',[200 100 700 400], 'MenuBar', 'none', 'Name', header, 'Resize', 'off',
'NumberTitle', 'off');

% Open 'filename' file... for reading...
fid = fopen(filename);
i = 1;                  % Subplot index on the figure...
while 1
   imagename = fgetl(fid);
   if ~ischar(imagename), break, end       % Meaning: End of File...
   [x, map] = imread(imagename);
   subplot(2,5,i);
   subimage(x, map);
   xlabel(imagename);
```

```
    i = i + 1;
 end

fclose(fid);
%-------------------------------------------------------------
```