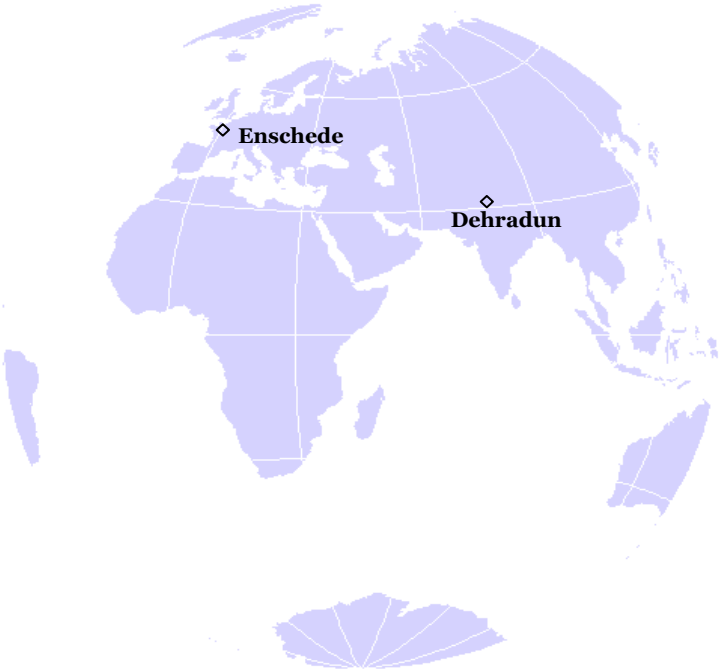
CONTENT-BASED SATELLITE CLOUD IMAGE RETRIEVAL

Deepak Upreti



M.Sc in Geoinformation Science and Earth Observation
Specialisation: Geoinformatics
April, 2011

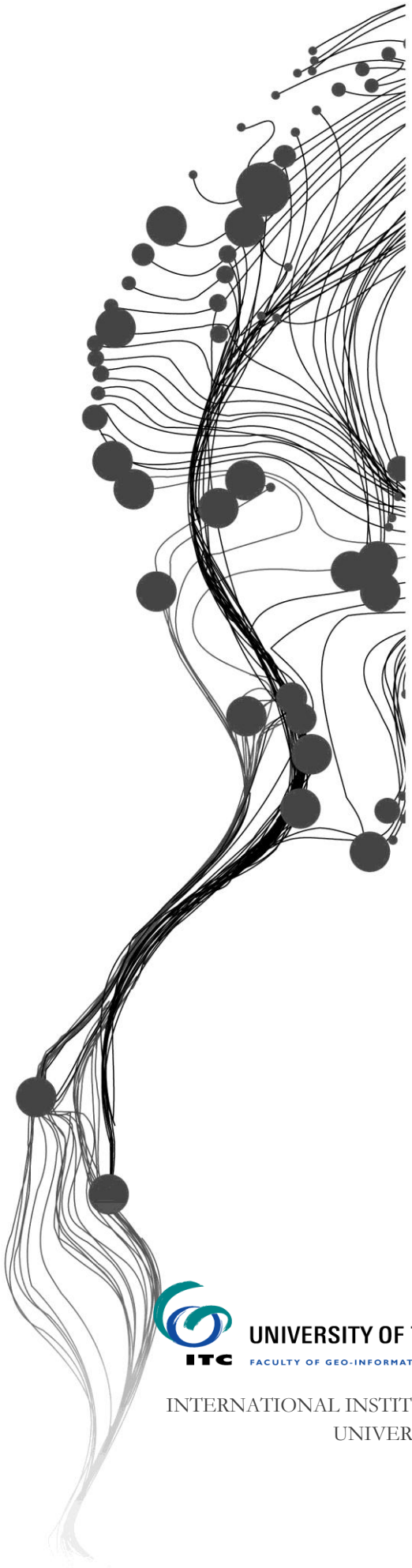
UNIVERSITY OF TWENTE.
ITC
FACULTY OF GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

Indian Institute of Remote Sensing
National Remote Sensing Centre
Government of India, Department of Space
ISO 9001-2008 Certified

CONTENT-BASED SATELLITE CLOUD IMAGE RETRIEVAL

DEEPAK UPRETI
March 2011

SUPERVISORS:
IIRS Supervisor: Dr. Sameer Saran
ITC Supervisor: Dr. Nicholas Hamm



CONTENT-BASED SATELLITE CLOUD IMAGE RETRIEVAL

DEEPAK UPRETI

Enschede, The Netherlands, March, 2011

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: M.Sc. Geoinformatics

SUPERVISORS:
IIRS Supervisor: Dr. Sameer Saran
ITC Supervisor: Dr. Nicholas Hamm

THESIS ASSESSMENT BOARD:
Prof. Ir. A. Alfred Stein (Chair)
Dr. R.D. Garg (External Examiner, IIT Roorkee)

UNIVERSITY OF TWENTE.
FACULTY OF GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

Indian Institute of Remote Sensing
National Remote Sensing Centre
Government of India, Department of Space
ISO 9001-2008 Certified

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION,
UNIVERSITY OF TWENTE, ENSCHEDE, THE NETHERLANDS
AND
INDIAN INSTITUTE OF REMOTE SENSING
DEHRADUN, INDIA

Dedicated to my loving mom and dad

ABSTRACT

The satellite cloud image is a valuable source of information in weather forecasting and early prediction of different atmospheric disturbances such as typhoons, hurricanes etc. Due to the increased number and resolutions of the Earth imaging sensors and image acquisition techniques, the satellite image data is growing enormously which makes it difficult to store and manage. The traditional image retrieval technique is inefficient in retrieving these images. Content-based image retrieval is an approach from data mining community which provides the solution of managing this huge quantity of data. In this research, a Content-Based Image Retrieval (CBIR) system has been developed using gray level, texture and shape as retrieval features from the satellite image repository. The system allows the user to search for an image on the basis of any of the three features alone or in combination by assigning weights to the features. The histogram approach is used to extract the gray level feature, texture feature is extracted using gray level co-occurrence matrix method and the shape feature is extracted using the morphological operations. The images and the extracted feature vectors are stored in the Oracle 10g database. Euclidean distance metric is used to compute the similarity between the images. The system is robust as it provides search based on the multiple features. The performance of the system was evaluated by analyzing the retrieval results using precision.

Keywords- Image retrieval, feature extraction, similarity computations, Euclidean distance, Content-based image retrieval.

ACKNOWLEDGEMENTS

I am heartily thankful to my IIRS supervisor, Dr. Sameer Saran, whose encouragement, guidance and support from the initial to the final stage enabled me to develop an understanding of the subject.

I am also thankful to my ITC supervisor, Dr. Nicholas Hamm, for providing me valuable support and suggestions on time, when required.

I wish to thank Dr. P.S. Roy (Dean, IIRS) and Mr. P.L.N. Raju (Head, GID, IIRS) and Mr. Shashi Kumar (Scientist- SC) for their extended support during research work.

I also thank my colleagues Tanvi Rajput, Preethi Balaji, Sourabh Pargal, Shreyes Shiv and Charles D. Richardson and Poonam Tiwari for helping me during my entire research period.

I also thank my brother Mr. Tarun pandey for supporting me in the development part of the research.

Lastly, I offer my regards to all of those who have supported me in any respect during the completion of the research.

Deepak Upreti

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation and problem statement

India has been traditionally vulnerable to natural disasters due to its unique geo climatic conditions. Cyclones, earthquakes, floods, droughts and landslides have been recurrent phenomena. About 8% of the total area is prone to cyclones. In the decade 1990-2000, an average of about 4344 people lost their lives and about 30 million people were affected by disasters every year [1]. Through the early prediction of the cyclones the causalities and the property loss can be minimized.

Earth observation (EO) is based on the recording of electromagnetic (EM) energy reflected back from the surface of the earth. Due to the increased number and resolution of earth observation imaging sensors, the acquired data volume and the information contents in the images have been significantly improved. This also increases the number of applications in which these data can be used, ranging [2] from weather forecasting over monitoring and managing of earth resources to navigation. With such a diverse set of applications, building a satellite image database becomes important [3]. The problems with creating satellite image database are the fast retrieval of useful images from remote sensing archives. To overcome this problem and to retrieve relevant images rapidly there is a strong need for highly efficient search tools for EO image databases.

Meteorological satellites operated by the Indian Space Research Organization (ISRO) have been collecting meteorological image data for over twenty five years with the launch of INSAT-1B in the year 1983. Furthermore, METSAT renamed Kalpana-1 on February 5, 2003 satellite uses three spectral bands for meteorological applications with the frequency of 30 minutes. These are visible band in the wavelength 0.55μm-0.75μm with the resolution of 2 km, thermal infrared band and water vapour band in the wavelength 10.5μm-12.5μm and 5.1μm-7.1μm respectively with the resolution of 8 km. This results in about 1500 Megabytes of image data per day and the manual search over this huge quantity of data is impractical.

The traditional satellite cloud image search method was based on the file name and the sensor parameters of every image. The disadvantages of this method are that it cannot describe the image contents such as cloud shape [4] and also leads to the inconvenience in retrieving images [5]. Clouds are the dynamic phenomena of the atmosphere so its effect on the climate is not known. By the better understanding of the clouds, better models can be developed for the prediction of different atmospheric disturbances such as typhoons, hurricanes, dust storms and etc. The better understanding of the clouds requires the study of the past images; a retrieval system can be used to study the historical patterns to study the current weather system [6].

There have been developed image retrieval systems for the retrieval of images from remote sensing archives. The retrieval of the images is mostly performed after the classification from the images [7] [8]. The disadvantage of this method is that the retrieval is based on the pre-extracted objects, to overcome this limitation new retrieval methods based on the independent features rather than on the classes are required [9].

Content-Based Image Retrieval (CBIR) is based on the low level visual features of the images these features are colour, texture and shape and spatial relation. Texture and shape are important features in the meteorological satellite images. Different types of clouds have different shapes. The types of clouds associated with the Tropical Cyclone (TC) are the cumulonimbus types of clouds. Also the different objects in the meteorological satellite images such as typhoons, hurricanes can also be described by the shape feature.

There have been developed image retrieval systems for the remote sensing image archival. The focus of this research is to develop a CBIR system for the retrieval of the TC images from the meteorological satellite archival that can allow the meteorologists to study the past weather system and understand the current weather system.

The architecture of a CBIR system is presented in Figure 1-1.

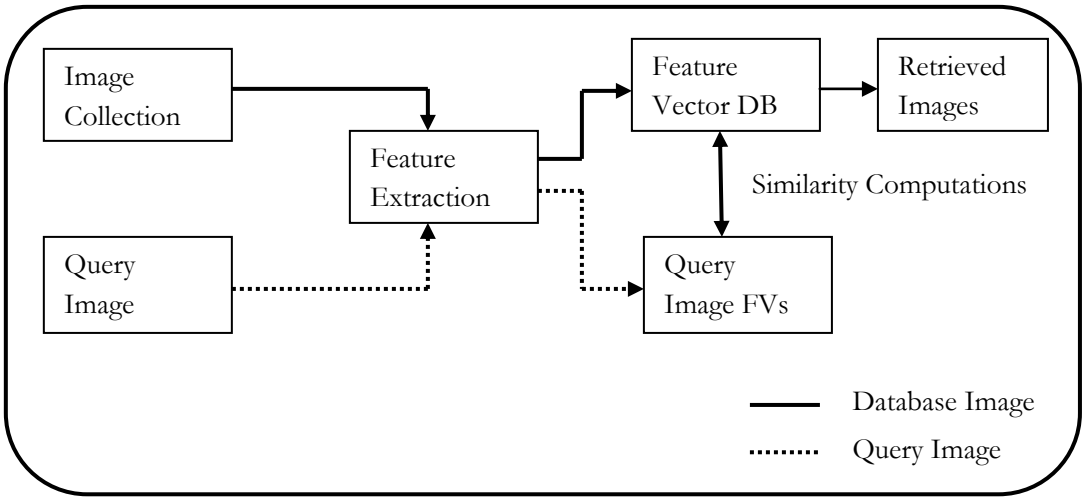


Figure 1-1: Architecture of a CBIR system

1.2. Research identification

1.2.1. Research objectives

The main objective of this research work is to develop a Content-Based Image Retrieval system for the meteorological satellite archival.
The main objective can be reached by defining following sub-objectives:

- To extract the features that characterize the satellite cloud image
- Storage of the images and feature vectors
- To compare the similarity between the query image and the stored images
- To evaluate the performance of the system

1.2.2. Research questions

- What methods will be used to extract the features?
- What will be the appropriate way of storing images and feature vectors?
- What will be the method of calculating the similarity between images?
- How can the images be ranked and retrieved?
- What will be the method to evaluate the performance of the system?

1.2.3. Innovation

The focus of this research is to develop a Content-based image retrieval system for the meteorological satellite archival using gray level, texture and shape feature.

1.3. Thesis structure

Chapter 1 explains the motivation and the problem statement behind this research work along with the derived research objectives and the research questions. Chapter 2 describes the basic components of a CBIR system and different approaches of image retrieval. The methodology of the research is presented in Chapter3. Chapter 4 explains the data set and the study area. The results and the discussions are presented in chapter5. Conclusions and recommendations are shown in the Chapter 6.

2. REVIEW OF LITERATURE

This chapter is dedicated to the understanding of the basic components of a CBIR system and how this approach by the informatics community can help in the analysis of TC. Section 2.1 discusses about the text based and the content based image retrieval approaches. The issues of the storage of the images and feature vectors are discussed in the section 2.2. Section 2.3 presents the applications of the CBIR approach in the satellite image retrieval.

2.1. Image retrieval

Due to the advances in the technology in various fields of image acquisition, computer science, image processing, information science and databases, the growth of the digital images and its management has become a research topic in different fields. The growth of the digital images on the web has created enormous data. Searching for an image in a vast collection of the images is a difficult task. There is a need of the efficient search tools to browse the large image collection. The two approaches of the image retrieval are the text based image retrieval and the content-based image retrieval.

2.1.1. Text based image retrieval

Image retrieval is a solution to search for an image in a large collection of images. The traditional approach of image retrieval is the text based approach. This approach relies on the textual descriptions of the images. Each and every image in the database is to be annotated with the textual keywords and the image search is based on these keywords. The limitations of the text based approach of image retrieval are lack of consistency of the text to describe the image contents and manual annotations of the images [10].

2.1.2. Content-based image retrieval

To, overcome the difficulties of the text based image retrieval, a new approach of the image retrieval, Content-based image retrieval (CBIR) has been a major research area for the users of different fields of computer vision, image processing and information retrieval. CBIR approach of image retrieval is based on image features rather on textual annotations to search for an image.

Feature is the representation of any distinguishable characteristic of an image [11]. The features in the images can be classified into three levels [12]. These are:

- **Low level features:** Colour, texture and shape are the primitive features of an image and can be extracted by information obtained at the pixel level.
- **Middle level features:** presence or arrangement of specific types of objects and can be extracted by collection of pixels that make an image.
- **High level features:** Includes the meaning associated with the combination of perceptual features. These features identify the meaning associated with the collection of pixels that make an object.

The basic components of the CBIR system are the feature extraction, storage of the images and the feature vectors and the similarity calculations between the images.

EO is based on the EM energy reflected back from the surface of the earth. This reflected energy is recorded by the sensor and is represented in the form of Digital Numbers (DN). Each pixel in each spectral band is represented by the DN value. So, each surface type is characterized by the vector of the DN values, also called spectral signatures [13]. Feature extraction in the remote sensing images refers to the computation of the signatures in the different spectral bands [14].

In remote sensing, spectral bands and the stacking of the spatial bands lead to the high dimension data [15], the problem with using all the information increases the computational load and time and is thus infeasible. The solution to the problem is reducing the dimensions of the data in such a manner that the information contained is not lost. Feature extraction is the process of reducing the dimensions of the image data by extracting the relevant features from the images. The features extracted from the images are smaller in size as compared to the original image data and are represented in the form of a fixed length real–valued multi-component. These components are called the feature vectors or signature [12] or descriptors [2] of an image.

Feature extraction is the most important module of a CBIR system. The efficiency of any CBIR system heavily depends on the features to be extracted. There are many features in the images ranges from pixels, regions, and objects to higher level descriptors of objects in the image [8], not all the features can describe the image. The selection of the appropriate features improves the efficiency and accuracy of a CBIR system. A review paper of 200 papers in [16] describes the different types of features used in a CBIR system.

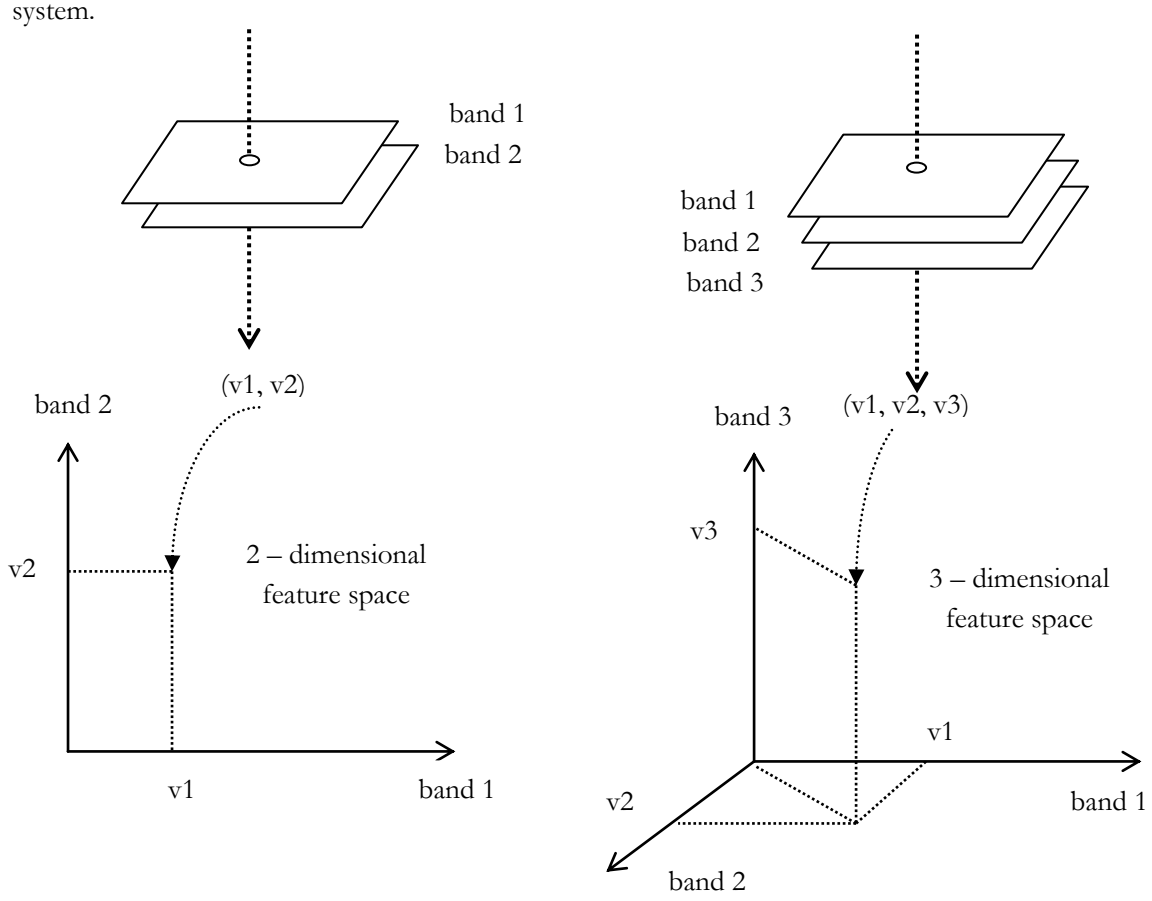


Figure 2-1: Feature vector plot in 2 and 3 dimensional graphs [17]

Database is used to store huge quantity of data such as image data. The advances in the database technology allow to store spatial data and to perform operations on these spatial data. Oracle with the release of Oracle Spatial 10g provides a tool Georaster that allow to the storage and management of the spatial data with several builtin functionalities. In CBIR systems, images and the feature vectors are mostly stored in the databases.

Similarity is a measure of distance between image features. Figure 2.2 shows the similarity of a query image to two images. Different similarity measures are described in literature, the commonly used in CBIR systems are Minkowski distance, Euclidean distance [18] and the quadratic distance metric.

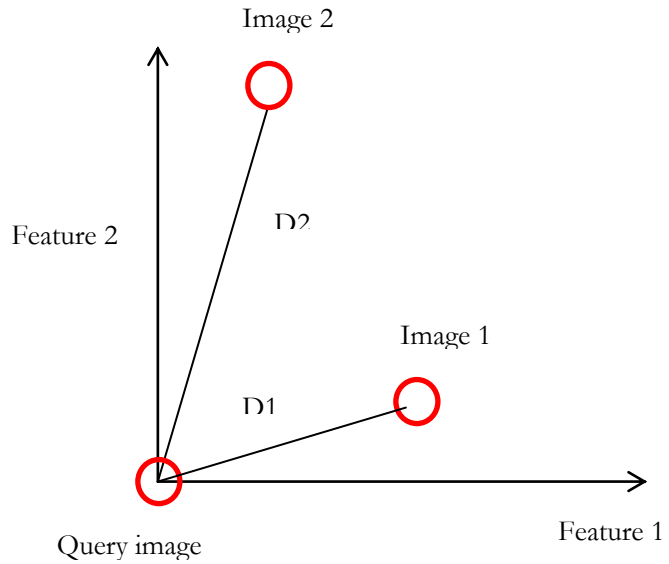


Figure 2-2: Similarity distance

The commonly used method of evaluating the performance of a CBIR system is Precision and Recall Graphs. The precision and recall are the measures of the effectiveness and robustness of the system [4].

2.1.2.1. CBIR based on low level image features

There has been developed many CBIR systems based on the low level image features in the different domains, some of these are of government use and some are commercial products. Some examples are crime prevention, security check, finger print scanning, medical diagnosis, trademark applications, archival of digital image libraries. Some CBIR systems based on low level image features are as:

- **Query by Image Content (QBIC):** A CBIR system developed by IBM, Almaden Research Centre, allows example images, user constructed sketches and drawings and selected colour and texture patterns combined with the textual annotations to query the large image databases [19].

- **Netra:** A CBIR system developed in [20] allows the user to search for an image using colour, texture, shape and spatial location. The system is implemented in Java language and tested with the 2500 colour images from the coral photo library. It allows user to formulate the query based on colour, texture, shape and spatial location or combination of them.

- **WAY-LOOK 4:** WhatAreYouLOOKing4 (WAY-LOOK4) system [21] retrieves images of interest based on the colour and the texture features. To speed up the retrieval, hash indexing is used. The similarity between the images is calculated in two steps, first the image is searched for the class to which it belongs and secondly, a linear search is used for that specific class to locate the image.

- **Visual Seek:** A CBIR system developed by the department of electrical engineering, Columbia University, allows matching of the images on the basis of colour and spatial locations. The image local regions and their colours, size and the spatial relationships are used to compare the images [22].

- **Content-Based Thermal Images Retrieval:** A CBIR system was developed in [23] for the retrieval of the thermal infrared images. The features of the image retrieval are temperature and spatial relations.

2.1.2.2. CBIR based on middle and high level features

The low level image features cannot completely characterize the images. Attempts have been made by the researchers to develop systems based on the middle level and high level image features. Middle and high level image features deal with the semantics of the images, such as an abstract objects and events etc. There remains the gap between the low level image features and the high level semantics of the images, called the semantic gap [16]. There have been developed methods to reduce this semantic gap. One technique of semantic based image retrieval is the relevance feedback; this technique brings the user in the loop and thus reduces the semantic gap [24]. The other techniques of the semantics based image retrieval are object-ontology, and the machine learning [25]. Some of these implementations are:

- A framework for an interactive CBIR system is presented in [26]. It uses the relevance feedback scheme to reduce the semantic gap, the gap between the human perception of images and the low level image features that describe the image. It uses the Radial basis Functions (RBF) neural network based learning machine.
- A new relevance feedback algorithm based on the multi modal model for the high level similarity is presented in [27]. The algorithm allows the users to query the databases more intelligently, which make it more robust and increases the retrieval performance.
- To reduce the semantic gap, an image semantic representation model (ISRM) is proposed in [28]. This model is an integration of the five parts these are image set, image feature set, image semantic set, semantic rule set and the semantic mappings, which describes the high level semantic content of the images. Based on the ISRM model and the Smooth Support Vector Regression (SSVR) theory; two algorithms of image retrieval example based and the text based were designed and implemented. The images database used contains 3,000 images of the natural scenery images with different semantics. Experimental results shows that the method is more robust as compared to the low level features based image retrieval.

2.2. Image storage

Images are the unformatted data; it can be stored in the file system as well as in the Relational Database Management Systems (RDBMS). The disadvantage of storing the images in file system is the management of the large amount of data. The images can be stored in the database as a Binary Large Object (BLOB) or can be stored externally as a BFILE with only a reference or a link to the external image content [29].

To overcome the disadvantage of file system storage, a CBIR system is proposed in [30]. To provide efficiency and effectiveness to the users this system extends the relational query in coupled with Adaptive Resonance Theory (ART2) neural network implemented in the Oracle PL/SQL.

The inner structure of databases in CBIR system is presented in [31]. This CBIR system contains three modules, a preprocessing module which is implemented in Matlab, a database module implemented in Oracle 10g and the user interface module created in Perl for world wide web (WWW) users. The oracle database includes a feature Georaster that is capable of managing large amounts remote sensing and GIS data and provides inbuilt functionality to manage these data.

A CBIR system for the medical image databases is implemented in [32], the Oracle database using the Oracle Intermedia is used as the storage of the images and the signatures generated from the images. For the easy implementation and the web access to the large image database on the web Java Server Pages (JSP) technology is used.

2.3. Domain specific CBIR systems

There have been developed CBIR systems for the retrieval of the satellite images. The following shows some systems for the retrieval of the satellite images from the archival.

Satellite cloud image is a valuable source of information and this information can be used in different fields. A CBIR system is presented in [3] uses the colour, texture and spatial relation characters as the main retrieval characters. Combining the image pre-processing results with the character extraction, the satellite cloud image retrieval was accomplished. The developed system is capable of fast and accurate retrieval of the images and can also provide weather forecast in the real time [5].

A prototype system for the retrieval of remote sensing images has been developed in [33]. The retrieval of the images is based on the colour and the texture features of the images. The colour moment's algorithm is used to extract the colour feature vector and the Gray level Co-occurrence Matrix (GLCM) method is used to extract the texture feature from the images. K-means clustering is used as it works well when the clusters are not well separated. Similarity matching is done using the Euclidean distance formula. The experiments were performed on four semantic categories mountains, vegetations, water bodies and residential areas of the LISS III multi-spectral images and the accuracy was 80%.

Contended-Based Satellite Cloud Image Processing and Information Retrieval (CBIPIR) system developed in [3] uses colour, texture and spatial relation as the main retrieval characters. The algorithm used to extract the texture feature is based on the GLCM, the five important texture measures out of 14 as suggested in [34] were extracted, and these are energy, entropy, contrast, correlation and local steady. The system outperforms well in providing the quick image search.

A shape based image retrieval system is presented in [35] for the retrieval of infrared satellite images. The retrieval is based on the three steps image pre-processing; region extraction which was performed using region growing segmentation and finally the region characterization was performed by polygonal approximation on the region shapes. To compare the polygon shapes polygon hashing method was adopted. Experiments were performed on the 56 Meteosat satellites images and to quantify the performance of the system, the hit and miss measure was used.

A three-layer recognition system is developed in [36] which include feature extraction module, the angle calculation module and the time warping module for the pattern matching of the current typhoon images with the Dvorak templates for the Tropical Cyclone (TC) early forecasting. The snake model [37][38] in the feature extraction module is used to extract the contour of the typhoon pattern. The angle calculation

module is used to assign the weights to the contour points depending on the distance from the TC eye. The angles between three successive points is calculated to reduce many point to point calculations and finally the dynamic time warping distance is used to calculate the similarity between the typhoon patterns.

An approach of shape based image retrieval is presented in [39] image processing is performed to extract the contour of an object in the image. This image processing includes the steps of image binarization, image erosion and the contour extraction. An ellipse is characterized by the semi major axis, semi minor axis and the orientation angle. These measures are considered as the feature vectors to retrieve the images of interest. An approach of particle swarm optimization (PSO) is used to compare the similarity between the images.

The past studies show the importance of an image retrieval system in this era of internet. Huge quantities of image data are being generated. No system exists that can be used to retrieve images in the different domains, the reason being the diversity of the images.

As highlighted by the authors, feature extraction process in an image retrieval system has always been a tedious task. The efficiency and accuracy of any retrieval system depends on the features to be extracted. There are many features in any remote sensing image, selecting all the features in an image is a not solution as not all the features contains much information about an image, the other disadvantage of selecting more features is that it increases the computational cost and time and decreases the performance of the system. The selection of more features often also leads to the curse of dimensionality; to avoid this dimensionality reduction algorithms such as Principal Component Analysis (PCA) are used.

Development of a CBIR system for the meteorological satellite archival will enable the image analysts and meteorologists to study the past weather system and will allow predicting the future weather system. By studying the past weather system better models can be developed which can be used for the early weather forecasting and also to estimate the intensity and track of the atmospheric disturbances as typhoons and hurricanes.

3. METHODOLOGY

The development of a CBIR system includes the integration of different components. Figure 3-1 shows the methodology of the work.

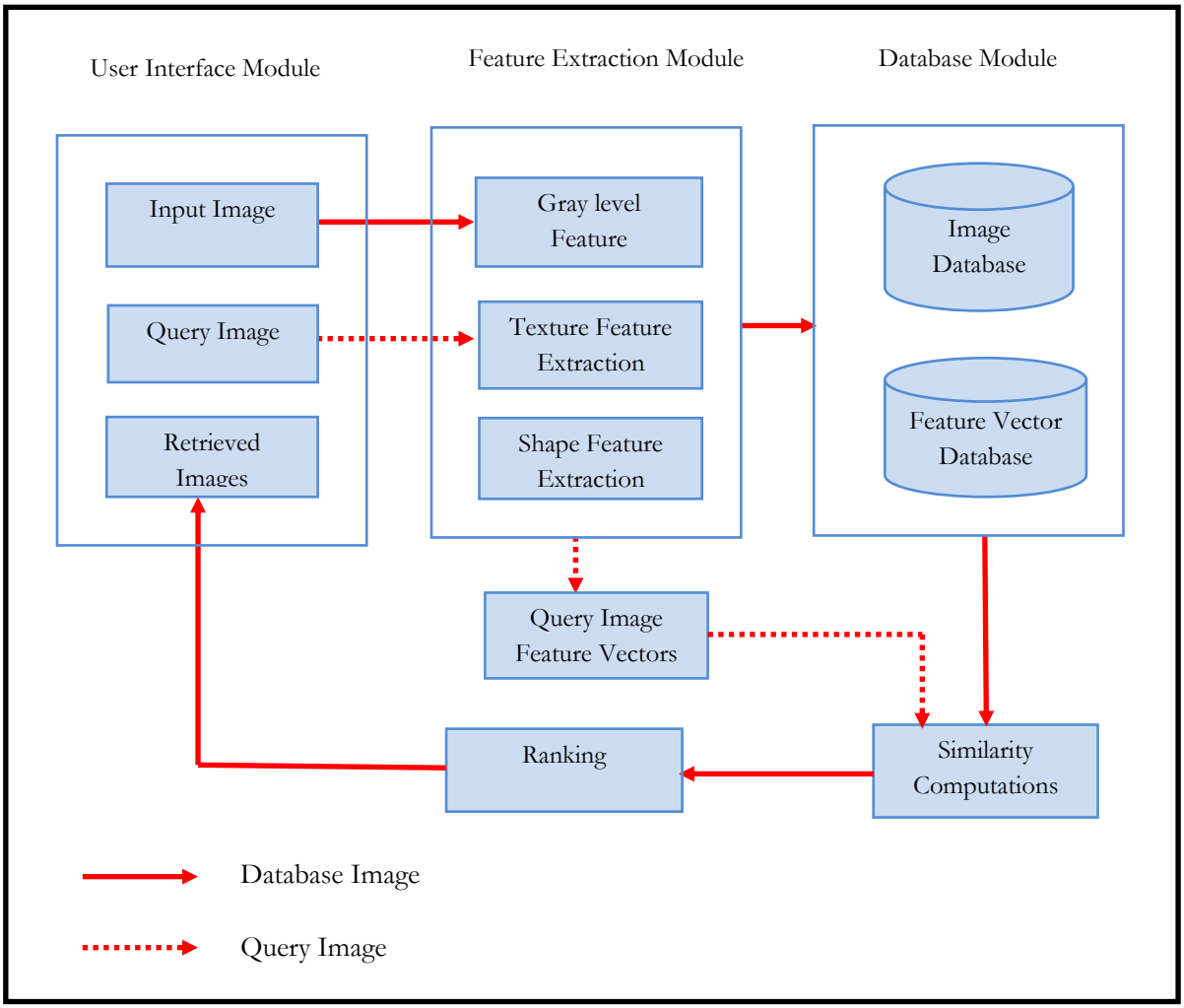


Figure 3-1: Work flow of the adopted methodology

3.1. Feature extraction

The most important module in the development of a CBIR system is the feature extraction module, because the efficiency and accuracy of any image retrieval system depends on the feature extraction module.

There are different types of clouds in the atmosphere; broadly categorized these are high level clouds, middle level clouds and low level clouds, these clouds can be better identified in the infrared images. In

infrared images high level clouds appear brighter than middle level and low level clouds. This gray level in the infrared satellite image is directly proportional to the temperature.

3.1.1. Gray level feature extraction

The histogram is a graphical representation of the occurrences of the gray levels or intensity in an image. The x-axis refers to the quantized gray level value and the y-axis refers to the frequencies of the gray levels [40].

Gray levels are extracted using the histogram approach of image retrieval. The histogram approach of image retrieval is mentioned in [41][42][43].

The gray level feature vector is extracted as:

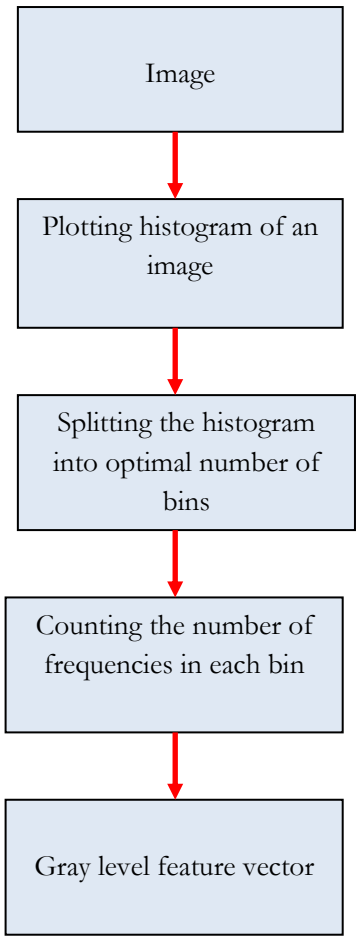


Figure 3-2: Gray level feature extraction method

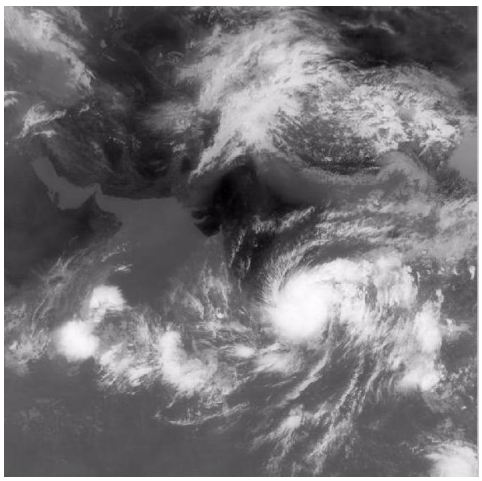


Figure 3-3: Infrared satellite image (K1VHR_17MAY2010_2300_L02_ASI)

© MOSDAC

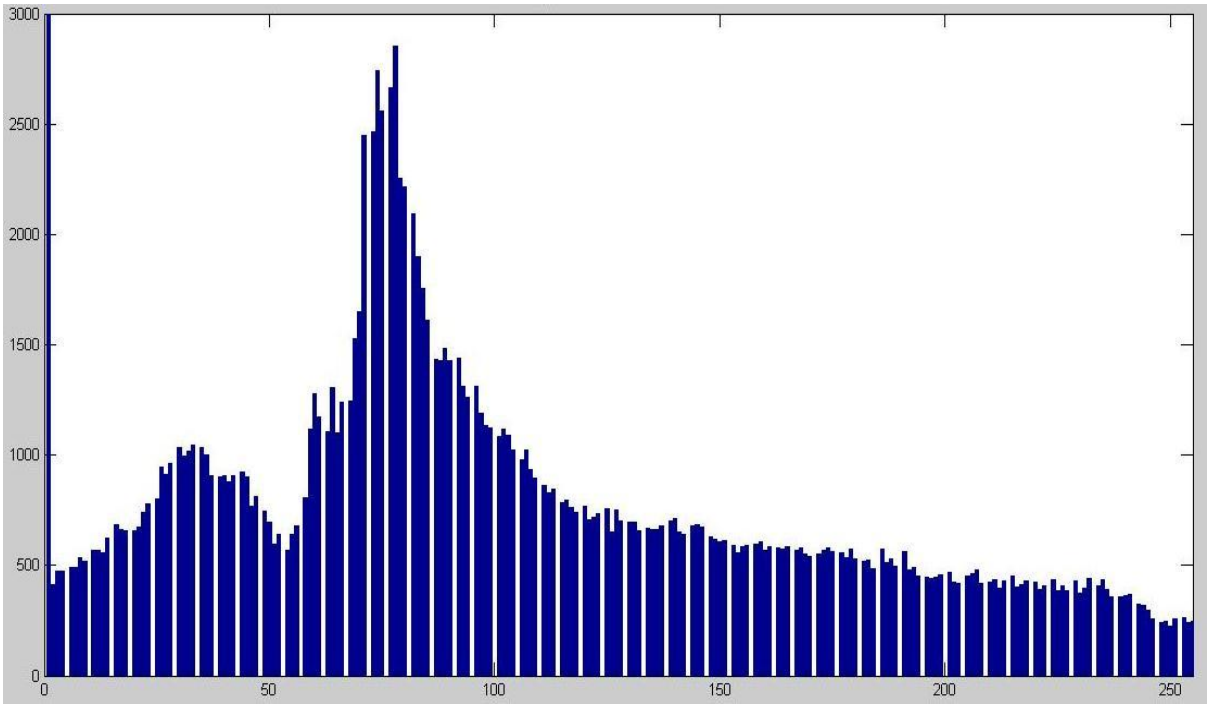


Figure 3-4: Histogram of an infrared satellite image

The image histogram shows the variations of gray levels from 0 to 255, these all values cannot be used as a feature vector as the dimension is too big to be stored or compared. The image histogram should be sampled into the number of bins to reduce the dimensions of the feature vector.

The sampling of the pixels into the optimal number of bins is necessary because very small bin width will represent the histogram in the form of spikes and will not contain much information to be used and the large bin width will increase the frequencies in each bin and will not be able to distinguish between different types of objects in the image and thus the retrieval accuracy will decrease.

Algorithm for selecting optimal bin width is mentioned in [44]. The steps of the algorithm are as follows:

- Divide the range of the histogram into N bins of bin width Δ, and count the number of pixels k_i in each bin from all n sequences that enter the i^{th} bin.
- Construct the mean and variance of the number of pixels $\{k_i\}$ as :

$$\bar{k} \equiv \frac{1}{N}\sum_{i=1}^{N} k_i \quad (1)$$

$$V \equiv \frac{1}{N}\sum_{i=1}^{N}(k_i - \bar{k})^2 \quad (2)$$

- Computing the cost function

$$C_n(\Delta) = \frac{2\bar{k} - v}{n\Delta^2} \quad (3)$$

➢ Repeating steps from i through iii and changing the bin width Δ to search for Δ_{min} that minimizes the cost function $C_n(\Delta)$.

The bin width Δ that minimizes the cost function is considered as the optimal bin width.
The bin width that were considered for the experiments were Δ = 4, Δ =8, Δ = 16 and Δ = 32, the minimum value of the cost function was found at Δ = 16.

The minimum value of the cost function was found at Δ = 16. So, the optimal bin width considered to extract gray level feature vector is Δ = 16. Figure 3-5 shows the optimal binned histogram of the image shown in Figure 3-3.

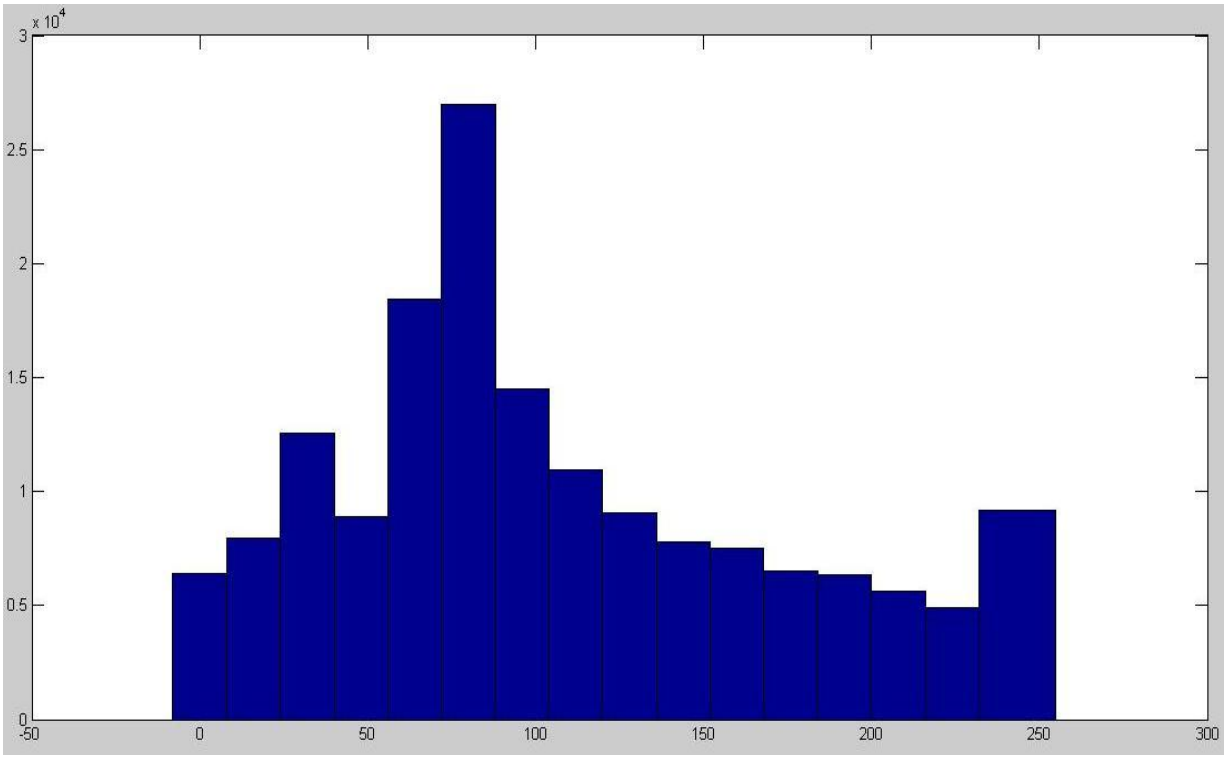


Figure 3-5: Optimal binned histogram of an image

The gray level feature vector is represented by the frequencies of the pixels in each of the bin of the histogram.

3.1.2. Texture feature extraction

There are different approaches of texture feature extraction as structural, spectral and statistical approaches. Statistical approaches use statistical techniques to compute the texture of an image. In this research due to significance and sake of simplicity, the statistical approach GLCM is used for satellite cloud image texture feature extraction.

In [34] 14 measures are suggested that can describe the texture of an image. Out of these texture measures Energy, Homogeneity, Contrast and Correlation are important to describe the texture of a satellite cloud image [3][5]. The Table 3-1 summarizes the above mentioned texture measures.

The method to extract the texture feature vectors is mentioned in Figure 3-6.

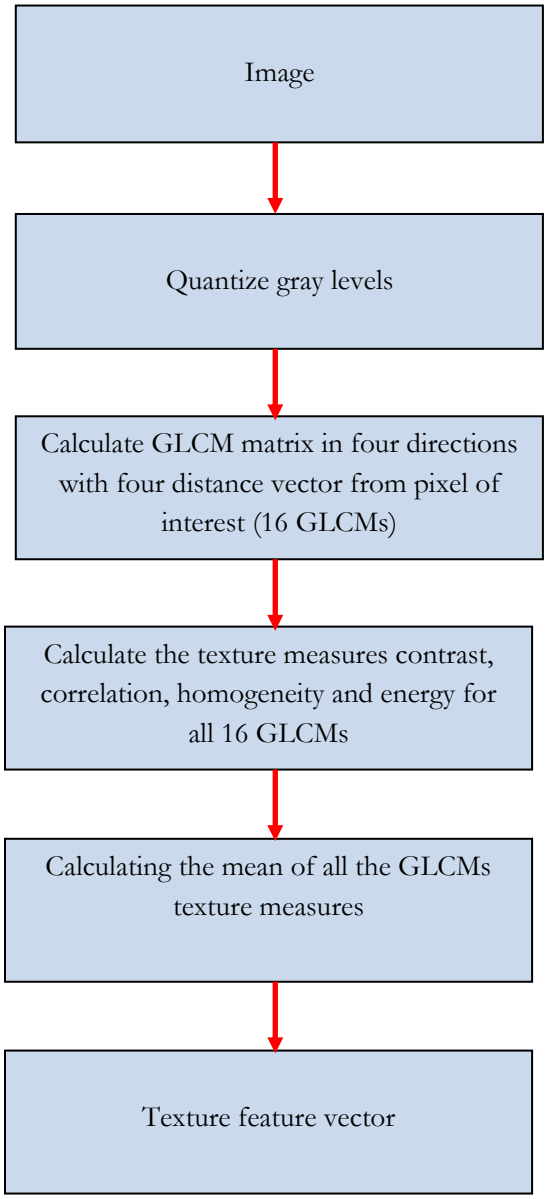


Figure 3-6: Texture feature extraction method

GLCM is a statistical method of examining texture that considers the spatial relationship of pixels in an image. This method characterizes the texture of an image by calculating the presence of the specific pairs and spatial relationship of the pixels in an image [45].

The radiometric resolution of Kalpana-1 satellite image is 10 bits. The gray levels are rescaled to 256 gray levels. In an image the number of possible gray levels affects the size of the co-occurrence matrix. Kalpana-1 satellite image if computed on 256 gray levels will generate the co-occurrence matrix of the size 256 × 256. The size of this co-occurrence matrix will increase the computational cost if the algorithm is applied to the rescaled image data. To, reduce the computational load the approach to be used is to quantize the gray levels in order to reduce the size of the co-occurrence [45]. The other advantage of the quantization of the gray levels is that it improves the statistical validity of the algorithm [46]. If the GLCM

is computed on the original 256 gray levels, there would be many cells filled with zero because of the appearance of the specific pair of gray levels. Having many zeros in the cells make this a very bad approximation. If the number of the zeros in the cells is reduced, the statistical validity is improved [46].

Table 3-1: Descriptors used in Gray level Co-occurrence Matrix [45]

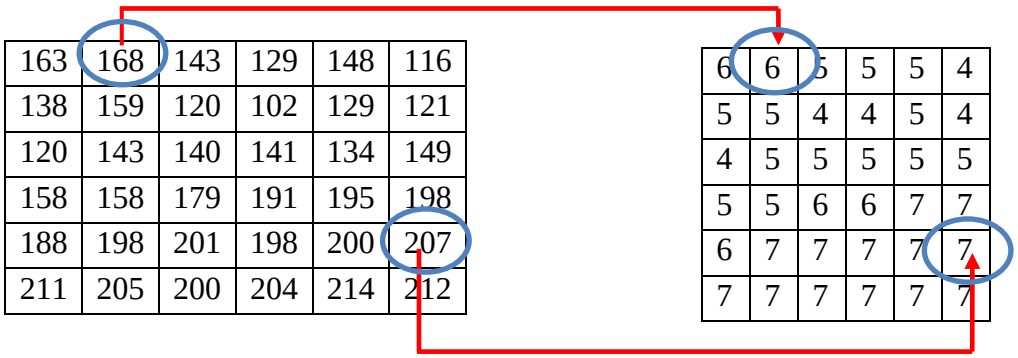
Descriptors	**Explanation**	**Formula**
Energy	A measure of uniformity (the sum of the squared elements in the GLCM) ranges from [0,1]	$\sum_{i=1}^{k}\sum_{j=1}^{k} p_{ij}^2$
Contrast	A measure of intensity contrast between a pixel and its neighbour	$\sum_{i=1}^{k}\sum_{j=1}^{k} (i-j)^2 p_{ij}$
Homogeneity	Measures the spatial closeness of the distribution of elements in co-occurrence matrix to the diagonal, its range lies between [0,1]	$\sum_{i=1}^{k}\sum_{j=1}^{k} \frac{p_{ij}}{1+\lvert i-j\rvert}$
Correlation	A measure of how a pixel is correlated to its neighbour over the entire image, its range lies 1 to -1	$\sum_{i=1}^{k}\sum_{j=1}^{k} \frac{(i-m_r)(j-m_c)p_{ij}}{\sigma_r\sigma_c}$ $\sigma_r \neq 0; \sigma_c \neq 0$

In this research the 256 gray levels are quantized into an 8 × 8 co-occurrence matrix [45], gray levels from 1 to 32 are considered as 1, then the next 32 as 2 and so on. Table 3-2 presents the quantization of the gray levels.

Table 3-2: Quantization of the gray levels

Gray levels of an image	**Quantized Gray levels**
1 - 32	1
33 – 64	2
65 – 96	3
97 – 128	4
129 – 160	5
161 – 192	6
193 – 224	7
225 - 256	8

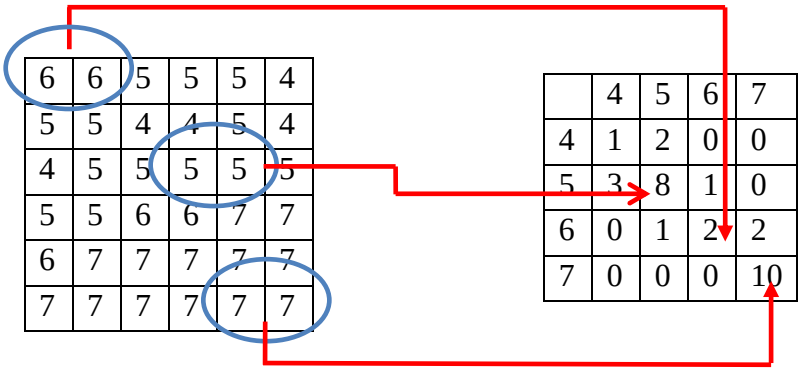
Figure 3-7 (a) shows an image with the original gray level values which lays in the range 0 to 255 and the Figure 3-7 (b) shows the quantized image as per the Table 3-2.



(a) Original image (b) quantized image

Figure 3-7: Quantization of the original image

Figure 3-8 (a) shows the quantized image and its transformation into the co-occurrence matrix; Figure 3-8 (b). The offset considered to compute the co-occurrence matrix is [0 1], that is the immediate pixel in the horizontal direction to the pixel of interest. The co-occurrence matrix is calculated by counting the number of times of the specific pairs of elements in the quantized image Figure 3.8 (a). It can be seen that [6 6] appear twice in the Figure 3-8 (a), so its value in the corresponding co-occurrence matrix is 2. The element [5 5] in Figure 3-8 (a) appear 8 times, so in co-occurrence matrix corresponding cell value is 8 and [7 7] appears 10 times, so in co-occurrence matrix, the cell contains the value 10.



(a) Quantized image (b) co-occurrence matrix

Figure 3-8: Transformation of image into co-occurrence matrix

There can be multiple GLCM generated from an image using different offsets and the distance from the pixel of interest as shown in Figure 3-8

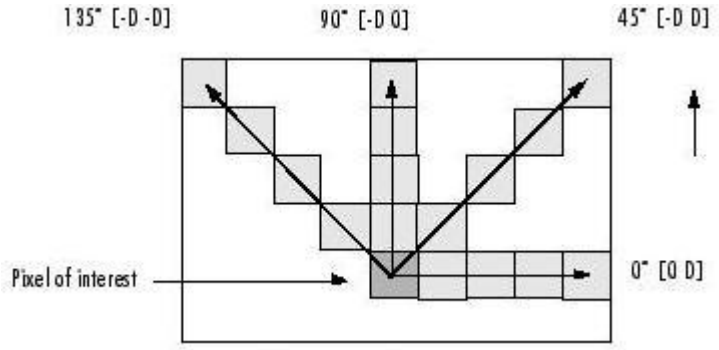


Figure 3-9: Spatial relationships of pixels [46]

Figure 3-9 illustrates the spatial relationships of pixels with different offsets and with the different distances from the pixel of interest. D in the Figure 3-9 represents the distance from the pixel of interest. To create multiple GLCMs for an image is necessary, because one GLCM cannot describe the texture of an image.

Figure 3-8 computes the texture of an image using the horizontal vector with one pixel distance to the pixel of interest. In this research using the 4 distance vector, in four directions the GLCM are created, that result in the 16 GLCM matrices which are used to calculate the texture measures. The following offsets are used to create the GLCM:

Offsets = [0 1; 0 2; 0 3; 0 4; -1 1; -2 2; -3 3; -4 4; -1 0; -2 0; -3 0; -4 0; -1 -1; -2 -2; -3 -3; -4 -4]

The texture measures contrast, energy, correlation and homogeneity for the 16 GLCMs are calculated by the formulae presented in Table 3-1. The obtained texture measures are 4 × 16 that is 64 in number. To reduce the dimensions and computational cost, the mean of these values is to be considered [47]. The mean can be calculated by the formula:

$$Mean = \frac{1}{N}\sum_{i=1}^{N} T_{mea} \qquad (4)$$

where N = 16, T_{mea} = Contrast, Correlation, Homogeneity, Energy.

The average value is calculated using equation (4) for the texture measures Contrast, Correlation, Homogeneity and Energy. These averages are used as the texture feature vector.

3.1.3. Shape feature extraction

Shape is an important feature of image retrieval; it is a local or region feature of an image and is closed to the human perception as humans can better recognize the shape of an object. There are two types of Shape descriptors, contour based shape descriptors and the region based shape descriptors, the former deals with the contour or boundary of an object in an image and the latter deals with the contour and the interior of the boundary also [48].

There are different approaches of TC image retrieval using shape feature including elastic graph dynamic link model [49], snake model [50], gradient vector flow snake model [36], Fourier descriptors [39], helix model [51] and etc.

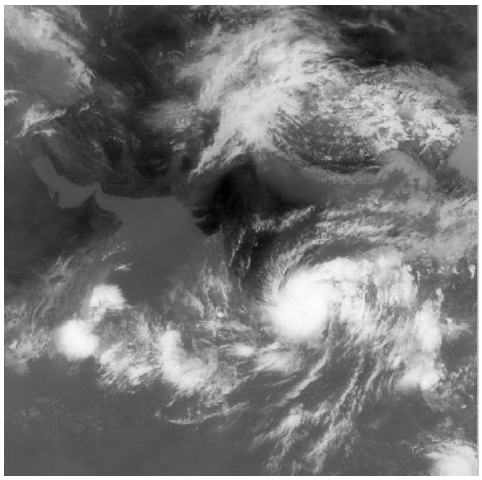


Figure 3-10: Infrared satellite image (K1VHR_17MAY2010_2300_L02_ASI)
© MOSDAC

The shape feature extraction is illustrated as follows:

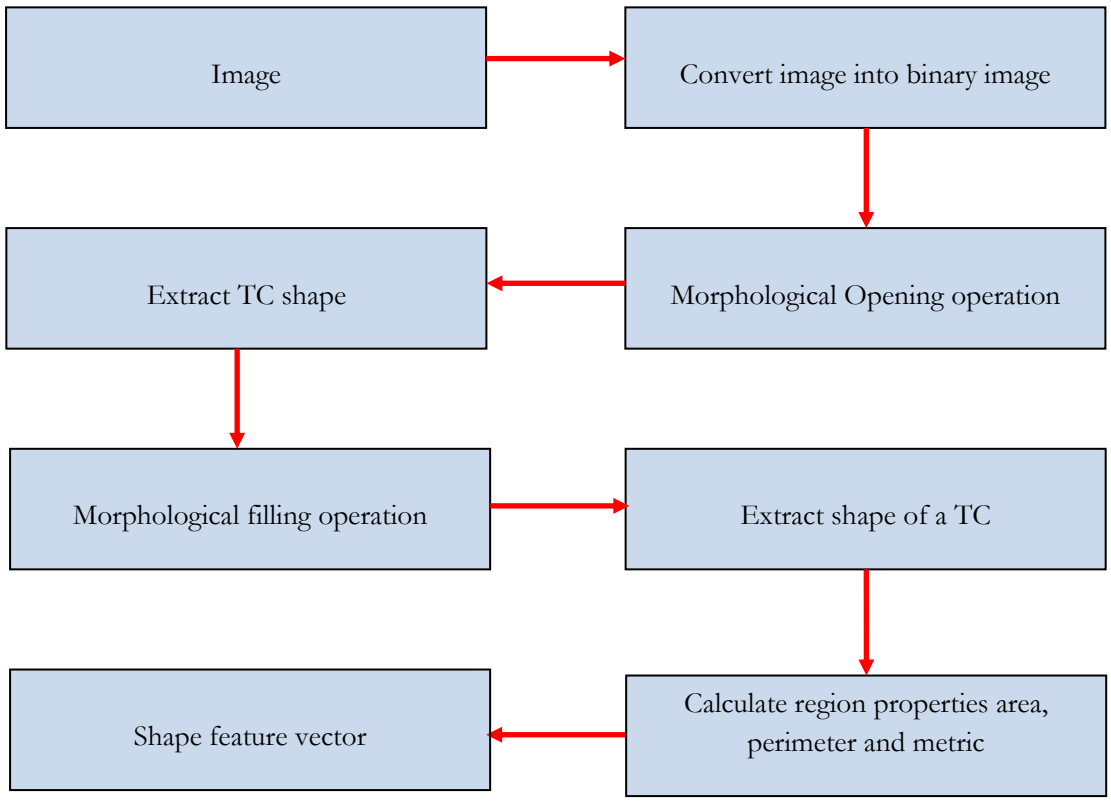


Figure 3-11: Shape feature extraction method

In this research, the image processing techniques and image morphological operations are used to extract the shape feature vector from the TC images. The infrared satellite image in Figure 3-10 is used to extract the shape feature vector. The shape of a TC cloud pattern can be separated better in the infrared images.

The image in Figure 3-10 in converted into binary image, to remove the noise. The foreground contains the TC cloud patterns and the clouds of the similar gray levels which are set to 1 and the background contains all the other clouds and is set to 0. Based on the experiments on the satellite images used in this research, the appropriate value of 0.94 is chosen as a threshold value to convert the satellite infrared image to the binary image. Figure 3-12 shows the converted binary image of the satellite image in Figure 3-10.

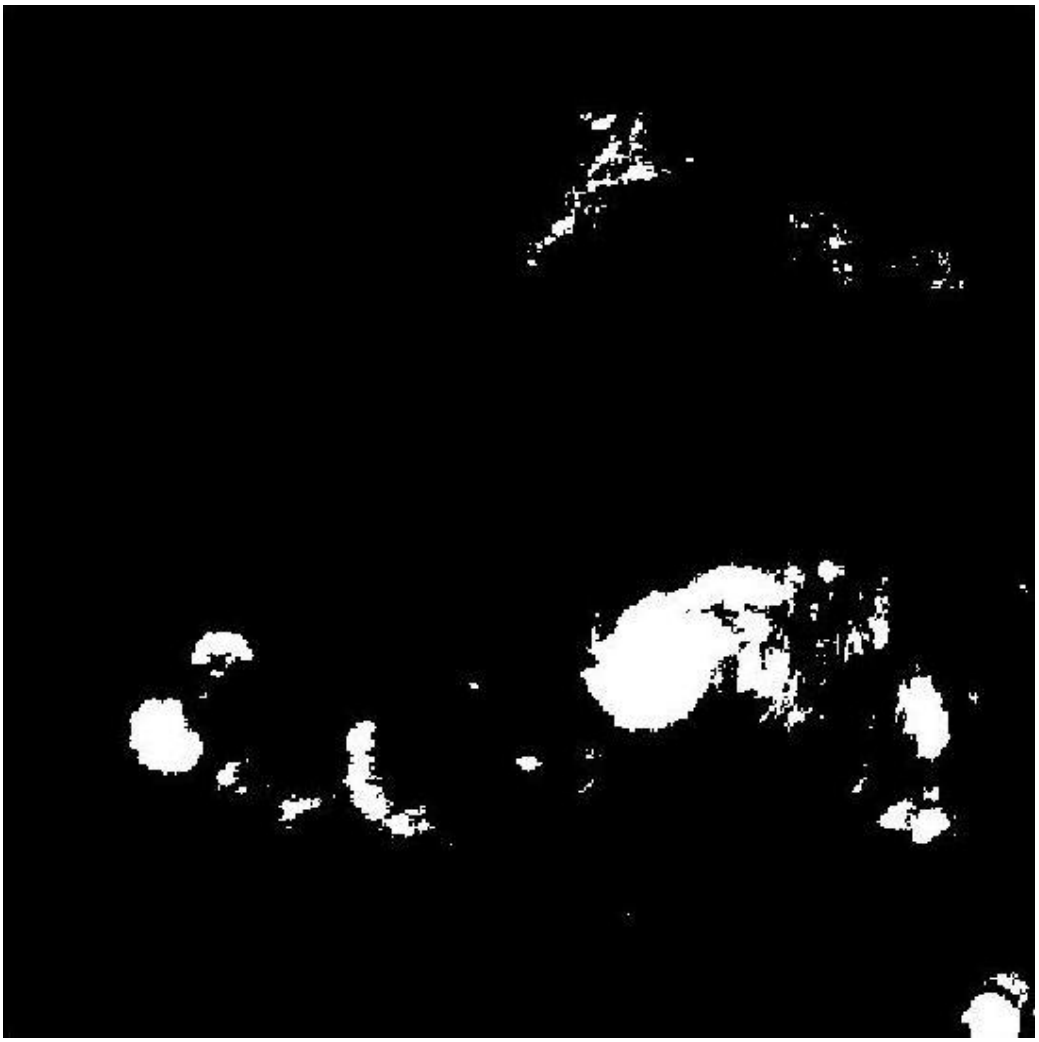


Figure 3-12: Binary image of the infrared satellite image using threshold value 0.94

Morphological operations produce an output image in which each pixel is based on the comparison of the input image and neighbourhood. A structuring element is a matrix containing only 0's and 1's that can take any arbitrary shape and size [52]. The following Figure 3-13 shows an example of a disc shaped structuring element and its origin.

The selection of the type of the structuring element depends on the objects to be identified in an image. In this research, the type of the object to be identified and extracted is elliptical in nature as TC is characterized by their surrounding spiral shaped cloud patterns. The type of the structuring element used is of disc type with size 3, based on the experiments on the original satellite images.

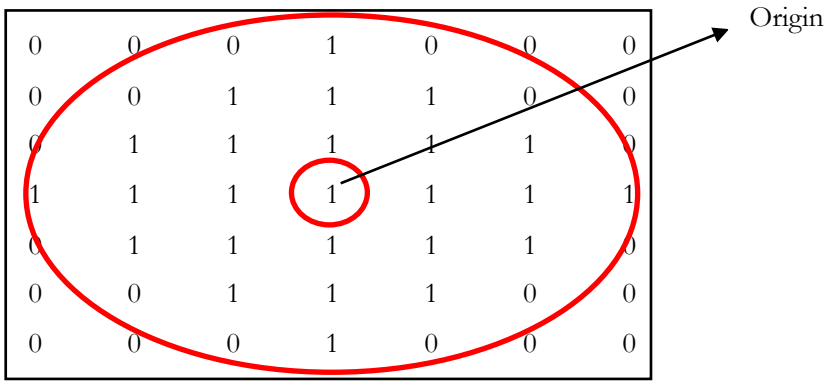


Figure 3-13: Disc shaped structuring element [52]

There are two basic types of the morphological operations erosion and dilation. The dilation operation adds pixels to the boundaries of the objects and the erosion operation removes the pixels from the object boundaries. The number of pixels added in the dilation operation and removed in the erosion operation from the object boundaries depends on the structuring element to be processed on the input image to produce the output image.

The erosion operation on the two sets A and B in Z^2 is defined in [45] as:

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \tag{5}$$

Equation (5) explains that set B, the structuring element has to be contained in the set A, which is equivalent to B not sharing any common elements with the background and can be expressed as:

$$A \ominus B = \{z \mid (B)_z \cap A^c = \emptyset\} \tag{6}$$

A^c is the complement of A and Ø is the empty set in equation (6).
The dilation operation on the two sets A and B in Z^2 is defined in [45] as:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} \tag{7}$$

Equation (7) is based on reflecting B about its origin and shifting this reflection by z. The dilation of set A by the structuring element, the set B is the set of all displacements, z, such that set $\hat{B}$ and set A overlap by at least one element. Equation (7) can also be written equivalently as:

$$A \oplus B = \{z \mid [(\hat{B})_z \cap A] \subseteq A\} \tag{8}$$

A is the set of image objects and B is the set of structuring element.

From the Figure 3.12 it can be seen that there is still noise in the image, all the noise has not been completely removed after converting satellite image in Figure 3-10 to binary image using an appropriate threshold value. The boundary pixels are also scattered, the boundary of a TC is not properly defined. The

morphological operation opening is used to remove the small objects from the image while preserving the shape and size of the larger objects in an image [52].

In [45] morphological opening operation is defined as:

$$f \circ b = (f \ominus b) \oplus b \quad (9)$$

f is an image and b is the structuring element.
Figure 3-14 shows the binary image after the morphological opening operation on the binary image in Figure 3-12.

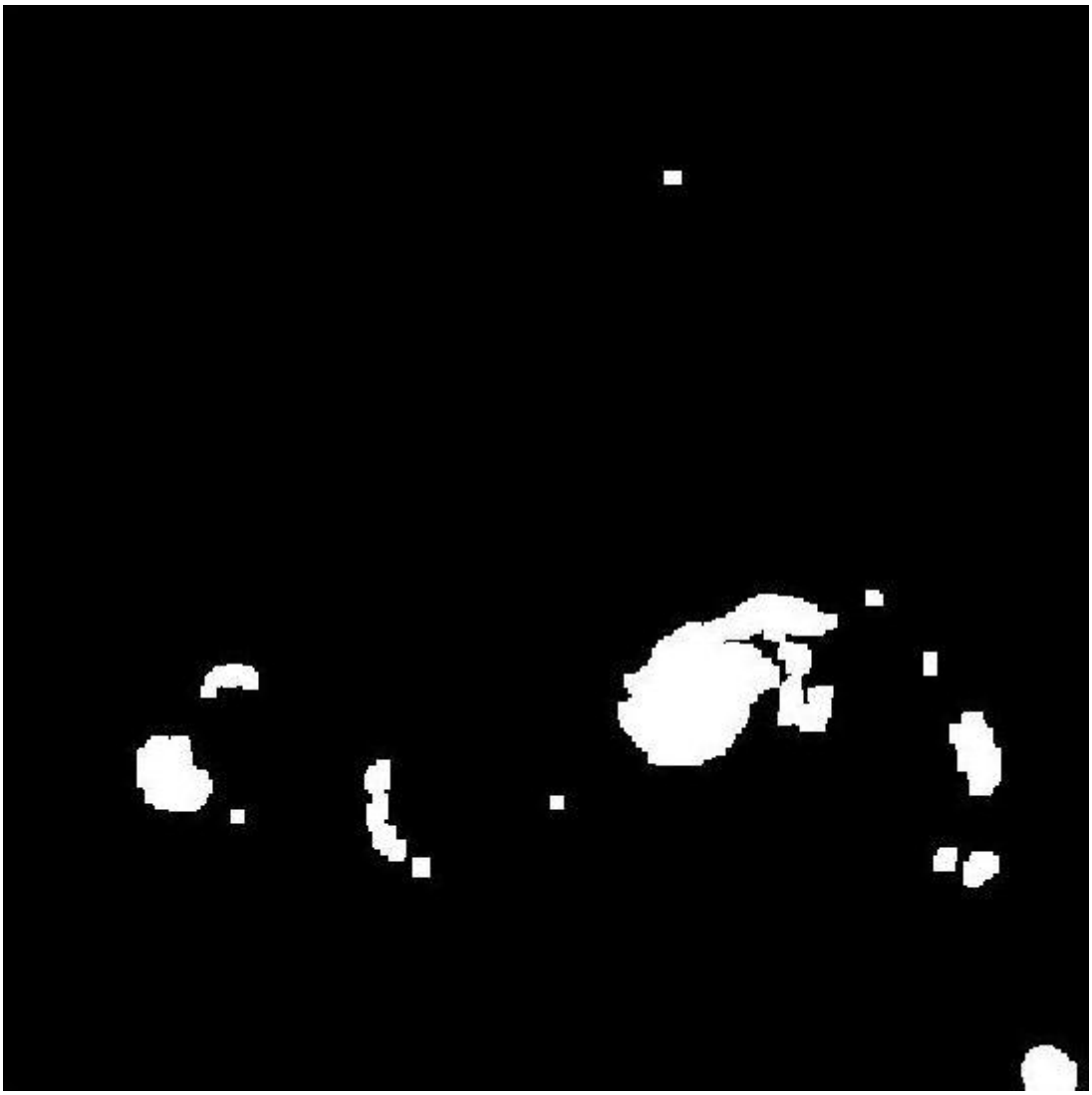


Figure 3-14: Binary image after the morphological opening operation

Due to experiments on the satellite images used, it is assumed that the TC contains at least 1500 pixels. All the other objects are removed that contains fewer than 1500 pixels. Figure 3-15 shows the binary image having objects that contains more than 1500 pixels.

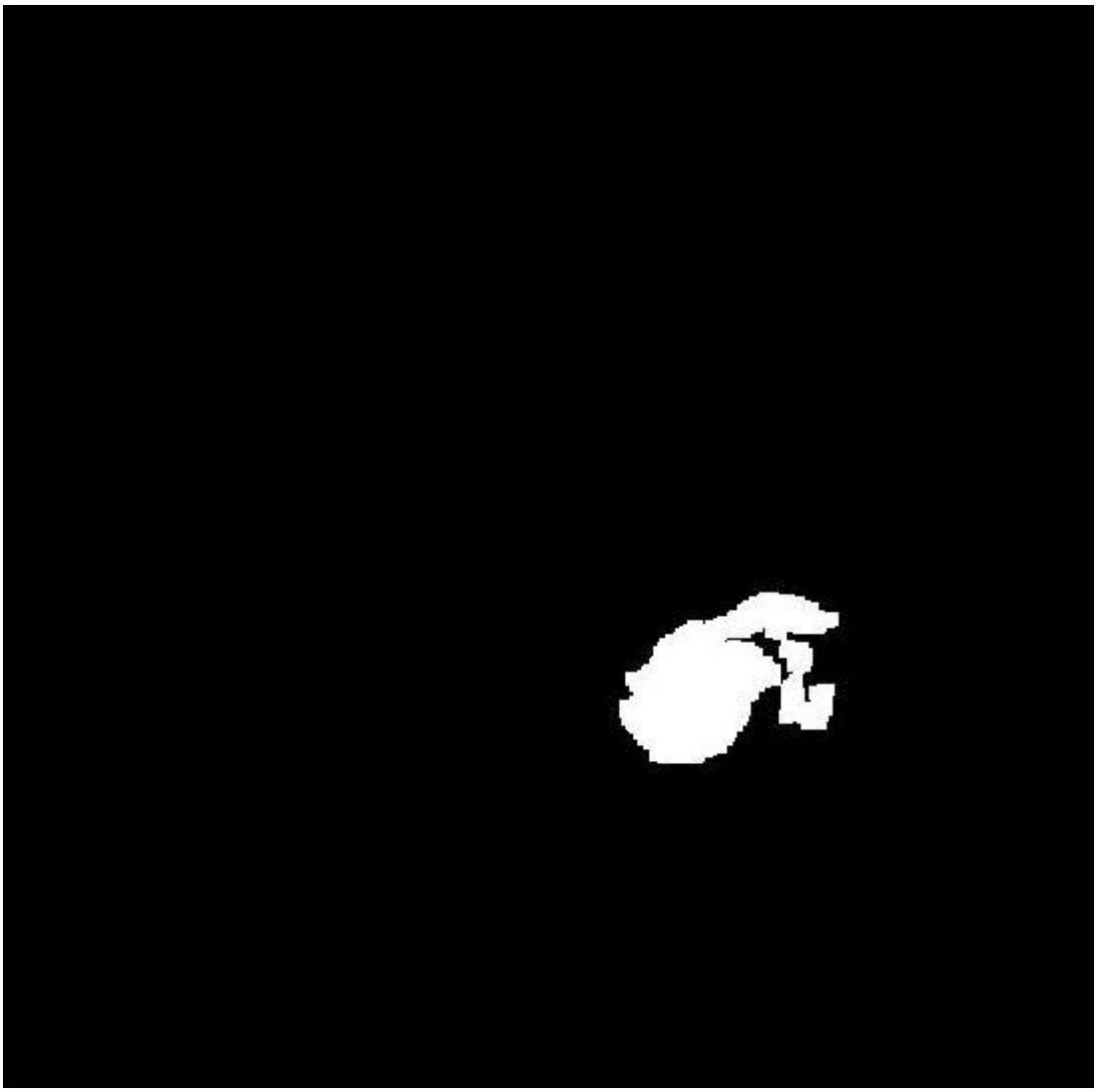


Figure 3-15: Extracted TC from the satellite image

There remain gaps in the TC as Figure 3-15; to fill the gaps inside the object the morphological operation filling is performed. The output of the filling operation is presented in Figure 3-16.

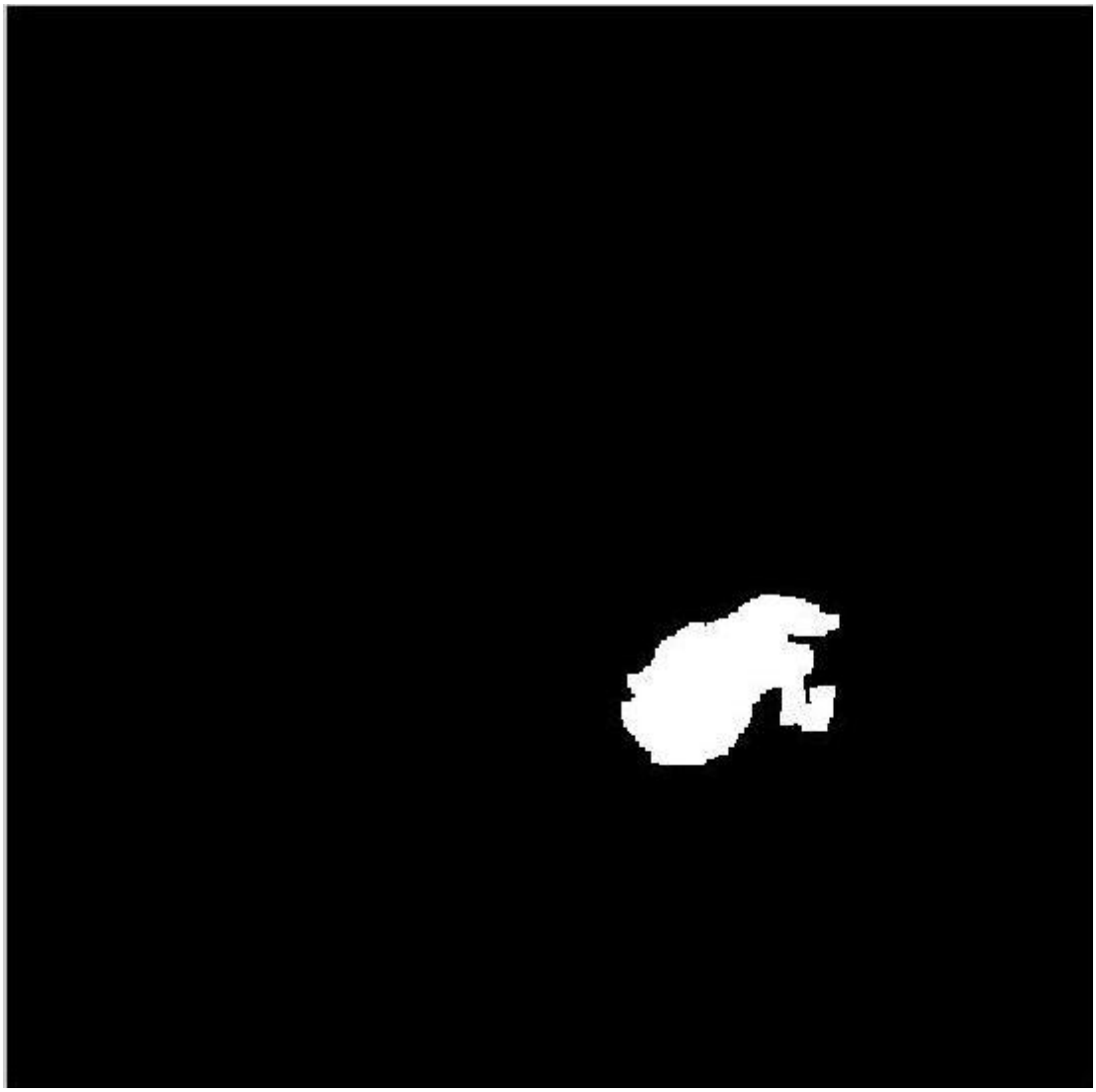


Figure 3-16: Extracted TC after the morphological filling operation

A TC is characterized by the surrounding spiral shape cloud patterns. The area, perimeter and the roundness are calculated from each of the TC pattern and stored in the database for the further processing. The shape of a TC here is described with the shape feature vector that consists of the three values for each of the TC satellite image. These values are area, perimeter and the roundness of the extracted TC from the infrared satellite image.

The boundary of the extracted TC is plotted in the Figure 3-17.

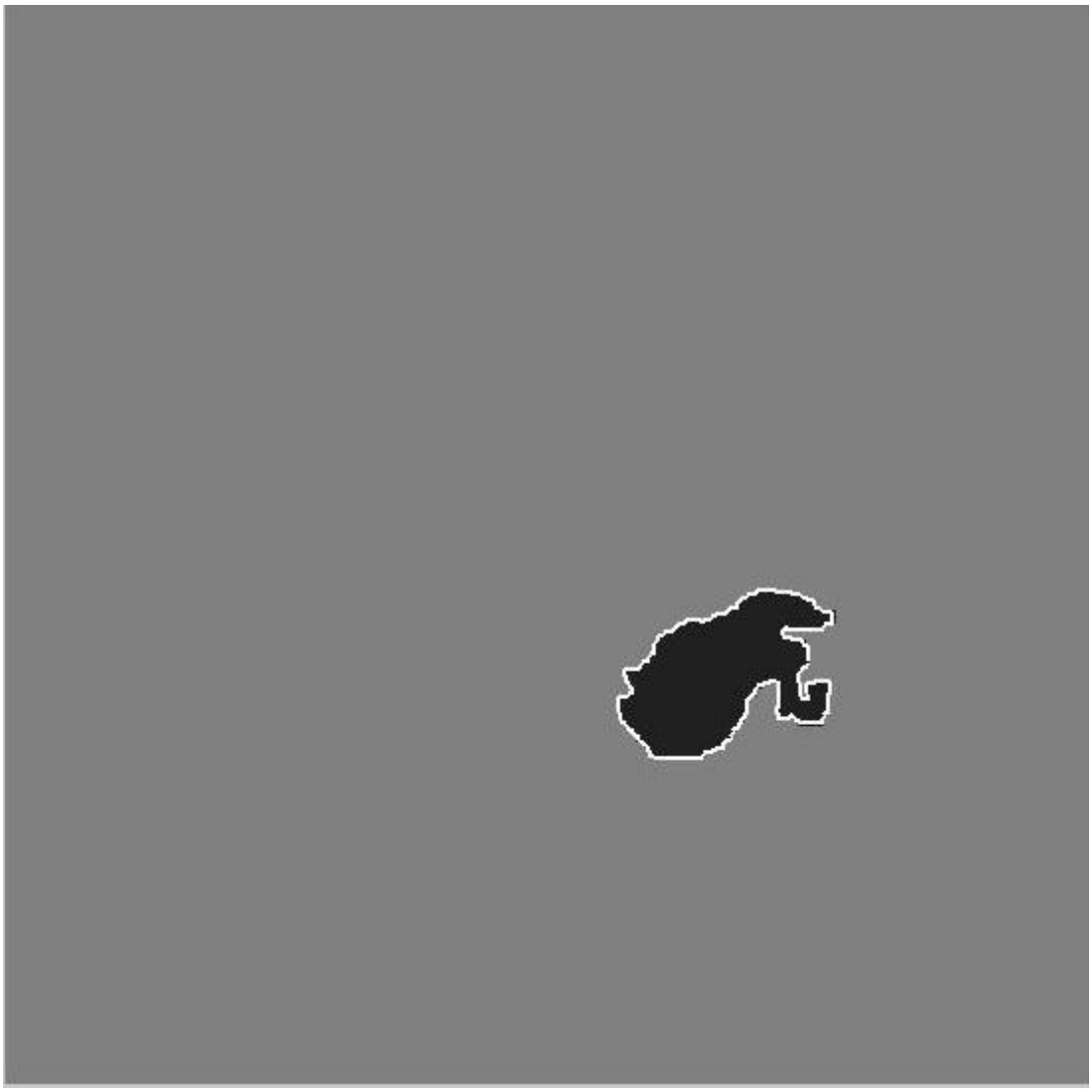


Figure 3-17: Boundary of the extracted T C

3.2. Database design

The extracted feature vectors and the images are stored in the Oracle 10g database. Four tables were created in the database. The first table includes two field's image name and the image. Second table contains image name and the sixteen fields to store the feature vectors of the gray levels. Third table stores the image name and the texture feature vectors and the fourth table includes the image name and the shape feature vectors of the images. The entity relationship diagram of the features and the images are shown in Figure 3-18.

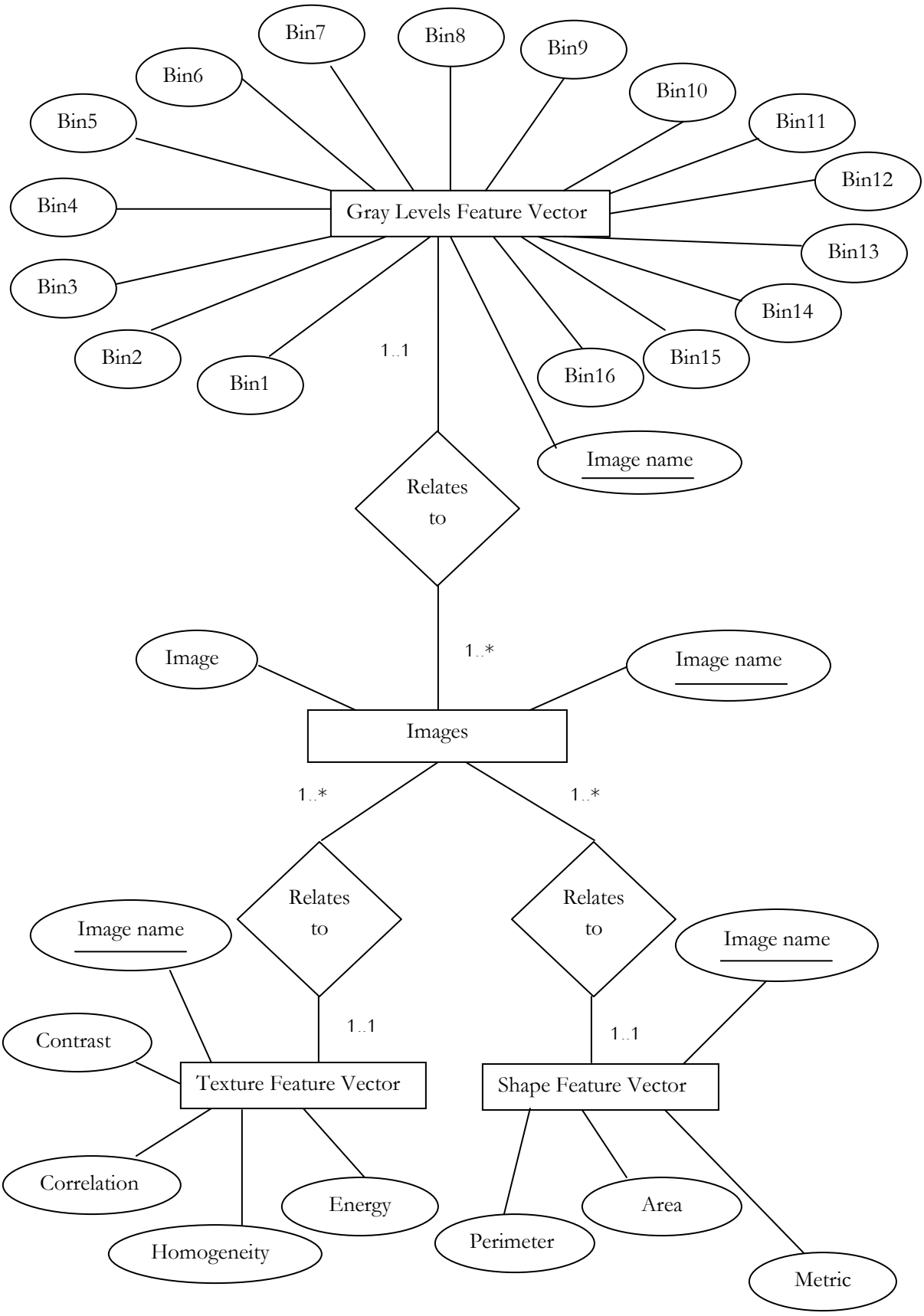


Figure 3-18: Entity relationship diagram for the system

3.3. Similarity computations

The similarity between the query image feature vectors and the feature vectors of the images in the database is computed using the Euclidean distance formula [18].

3.3.1. Gray levels similarity calculation

Euclidean distance formula for comparing the similarities between query image and the database image is used in [41]. The similarity between the gray levels feature vector of the query image and the gray levels feature vectors of the images in the database is calculated by the formula mentioned in the equation (10).

$$Sim_{gr}(QI, DBI) = \sqrt{\sum_{n=1}^{16} (QI_n - DBI_n)^2} \tag{10}$$

QI is the query image and the DBI is the database image.

3.3.2. Texture similarity calculation

In [53], the similarity between the images are calculated using the Euclidean distance The similarity between the texture feature vector of the query image and the texture feature vectors of the images in the database is calculated by the formula in the equation (11)

$$Sim_{tex}(QI, DBI) = \sqrt{\sum_{n=1}^{4} (QI_n - DBI_n)^2} \tag{11}$$

QI is the query image and the DBI is the database image.

3.3.3. Shape similarity calculation

Euclidean distance is the standard formula to compute the similarities between the images. Different shape feature extraction techniques extract the shape of an object and then represent the shape by a set of values, called the shape feature vector. The similarity between the shape feature vectors is computed using the Euclidean distance formula. In this research, the similarity between the shape feature vector of the query image and the shape feature vectors of the images in the database is calculated by the formula in the equation (12)

$$Sim_{shape}(QI, DBI) = \sqrt{\sum_{n=1}^{3} (QI_n - DBI_n)^2} \tag{12}$$

QI is the query image and the DBI is the database image.

3.3.4. Final similarity calculation

The user of the system can retrieve the images based on the weights assigned to the particular features. The final similarity is calculated as [18] in equation (13)

$$Sim\,(QI, DBI) = W_{gl} * Sim_{gl} + W_{tex} * Sim_{tex} + W_{shape} * Sim_{shape} \quad (13)$$

Where $W_{gl} + W_{tex} + W_{shape} = 1$

QI is the query image and the DBI is the database image.
W_{gl} = weight assigned to the gray level feature
W_{tex} = weight assigned to the texture feature
W_{shape} = weight assigned to the shape feature
Sim_{gl} = gray level similarity
Sim_{tex} = texture similarity
Sim_{shape} = shape similarity

3.4. Ranking and retrieval of images

The images are sorted and ranked on the basis of the weighted similarities values. The steps to retrieve the images are as follows:

- For all similarities values, check similarity value (previous) < similarity value(next)
- If similarity value (previous) < similarity value(next)
    - Swap similarity value (previous) and similarity value(next) as:
        - Assign similarity value(next) to temporary variable
        - Assign similarity value (previous) to similarity value (next)
        - Assign temporary variable to similarity value (previous)
- Swap image names according to the similarity value
- Retrieve initial 12 sorted similarity values and their corresponding images.

3.5. User interface design

User interface is a middleware between the user and the database. It allows user to enter query and display the similar images that are in the database.
In this research user interface is created in the Matlab Graphical User Interface Development Environment (GUIDE). The flow of the user interface is presented in the Figure 3-19.

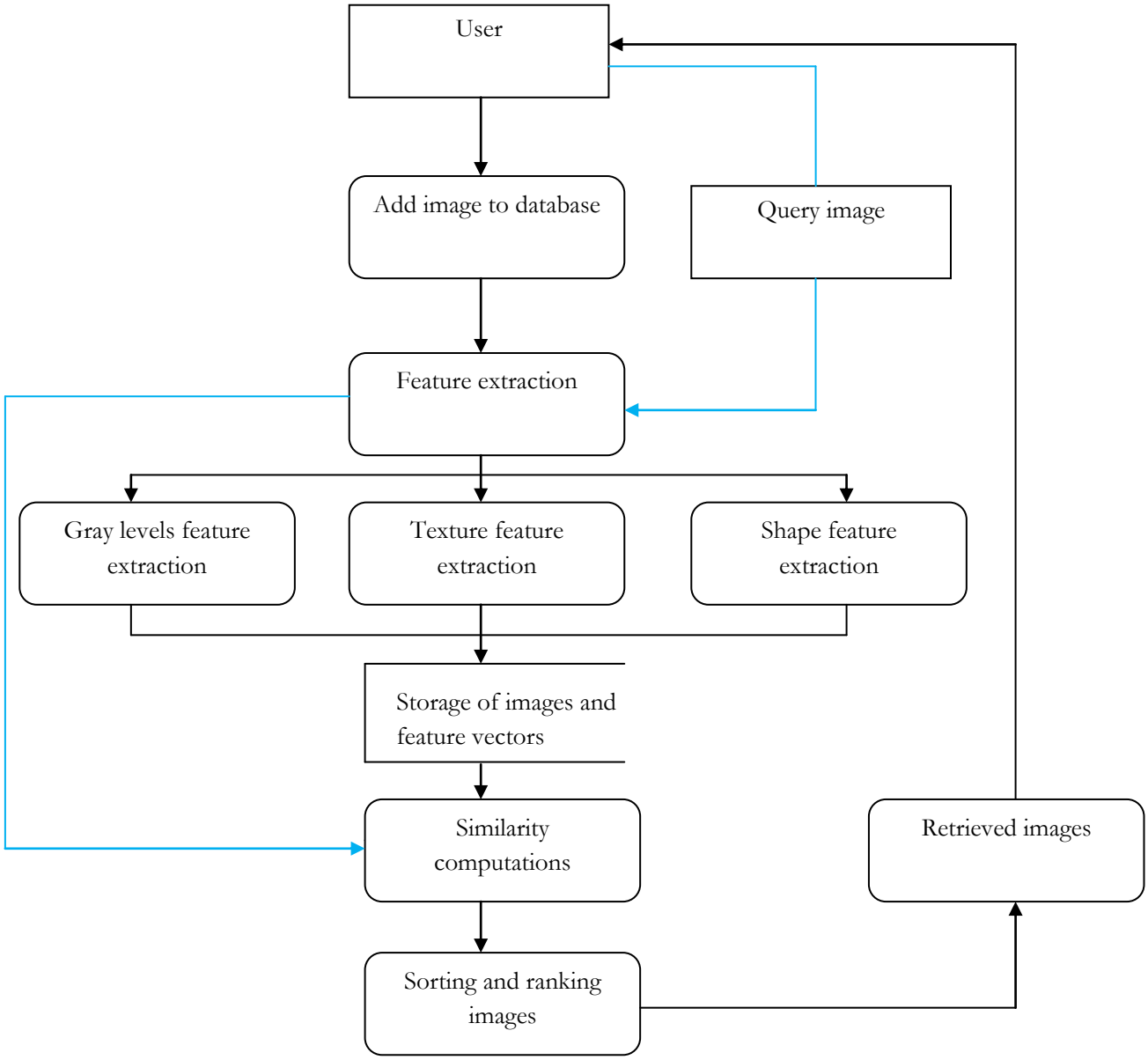


Figure 3-19: Data flow diagram of a user interface

3.6. System performance and evaluation

The standard measures of evaluating the performance of a CBIR system are precision and recall. This method is used to evaluate the performance of the system in [4] for the typhoon image retrieval. In this research, the performance of the system is evaluated on the basis precision in the retrieving the images.

3.6.1. Precision

Precision is defined as the ratio of the number of the relevant images retrieved to the total number of the images retrieved. It is a measure of the accuracy of the retrieval [48] and can be defined as [54][7][4]:

$$Precision = \frac{Number\ of\ relevant\ images\ retrieved}{Total\ number\ of\ images\ retrieved} \quad (14)$$

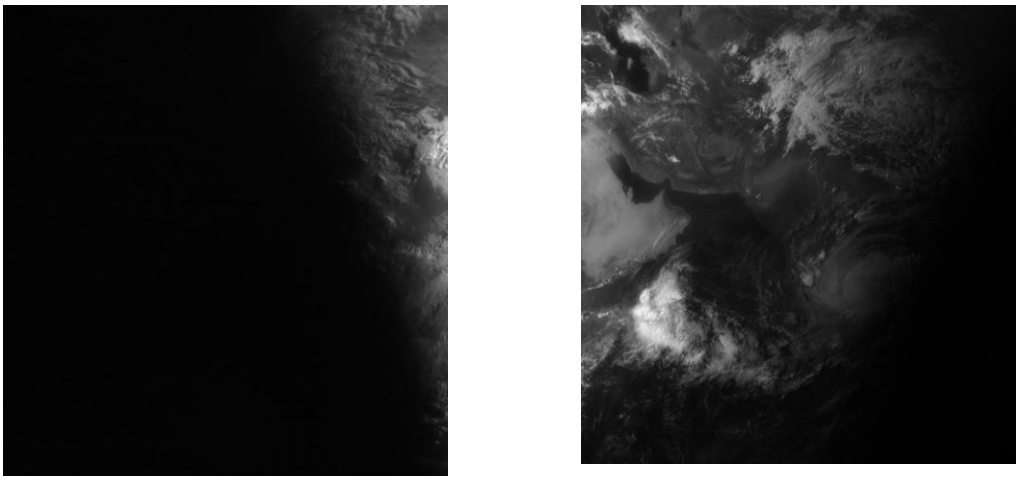
4. STUDY AREA AND MATERIALS

4.1. Indian meteorological satellite kalpana-1 satellite images

Indian meteorological satellite Kalpana- 1 was launched in the year 2002, it uses three spectral bands for meteorological applications. These are visible band, thermal infrared band and water vapour band in the wavelengths 0.55μm-0.75μm, 10.5μm-12.5μm and 5.1μm-7.1μm respectively with the resolution of 2 km in the visible band and 8 km in the thermal infrared and the water vapour band. Examples of these images are described and shown in the sections 4.1.1, 4.1.2 and 4.1.3.

4.1.1. Visible band images

Kalpana- 1 uses the wavelength of 0.55μm-0.75μm in visible band. This range is used to identify different reflecting objects such as clouds, deserts and snow in the images. The resolution in this band is 2 km. The disadvantage of visible band imagery is that, it is not available in the night. The example images of this band are shown in Figure 4-1.



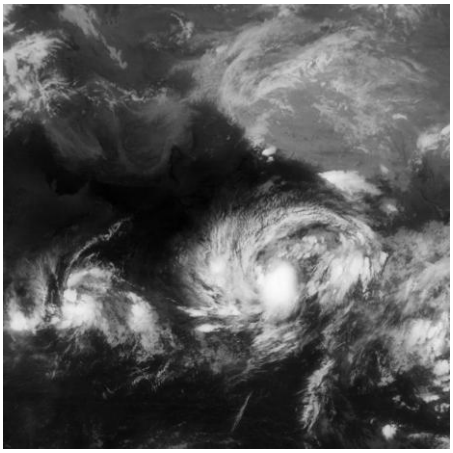
(a) May 18, 2010, 11.00 pm. © MOSDAC

(b) May 18, 2010, 11.30 am © MOSDAC

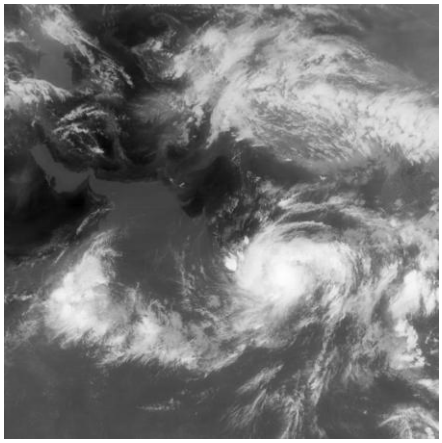
Figure 4-1: Visible band image

4.1.2. Thermal infrared band images

The infrared images capture the temperatures of the different objects in an image. There are different types of clouds in the atmosphere; these different types of clouds can be differentiated in the infrared imagery. The low level clouds have the low gray-level value and the high level clouds have the high gray-level value and appear brighter than the low level clouds. Kalpana-1 satellite uses the wavelength 10.5 μm-12.5 μm with the resolution of 8 km in the infrared band. The images of the infrared band are shown in Figure 4-2



(a) May 18, 2010, 11.00 pm
© MOSDAC

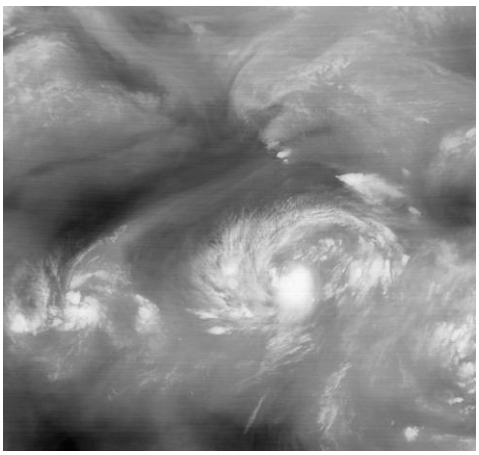


(b) May 18, 2010, 11.30 am
© MOSDAC

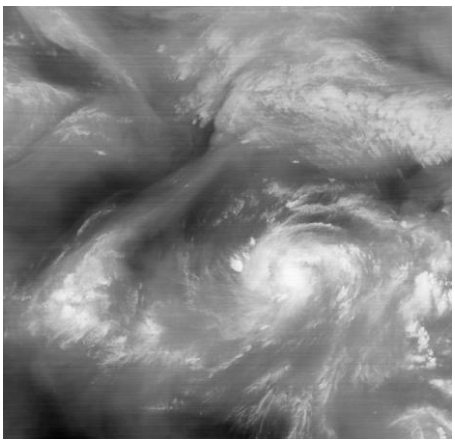
Figure 4-2: Infrared band image

4.1.3. Water vapour band images

Water vapour imagery is useful in the detection of the moisture content in the upper atmosphere. Brighter values indicate more moisture and the darker values represent low moisture content in the atmosphere. Kalpana-1 satellite uses the wavelength 5.1 μm-7.1 μm of the electromagnetic spectrum with a resolution of 8 km for the detection of the water vapour content in the atmosphere. The two images taken at different is shown as an example in Figure 4-3



(a) May 18, 2010, 11.00 pm
© MOSDAC



(b) May 18, 2010, 11.30 am
© MOSDAC

Figure 4-3: Water vapour band image

4.2. Study area and data used

Table 4-1 shows the information about the Indian meteorological satellite Kalpana-1.

Table 4-1: Satellite Information

Satellite	Kalpana-1
Sensor	VHRR
Frequency	Half hourly
East longitude	110
West longitude	40
North latitude	50
South latitude	10

The geo-climatic conditions of India make it prone towards the natural disasters. Cyclone is the mostly occurring atmospheric disturbances in India. The east coast of India is more affected by the cyclonic disturbances. The study area of this work is the east coast of India, the Bay of Bengal as shown in Figure 4-4.

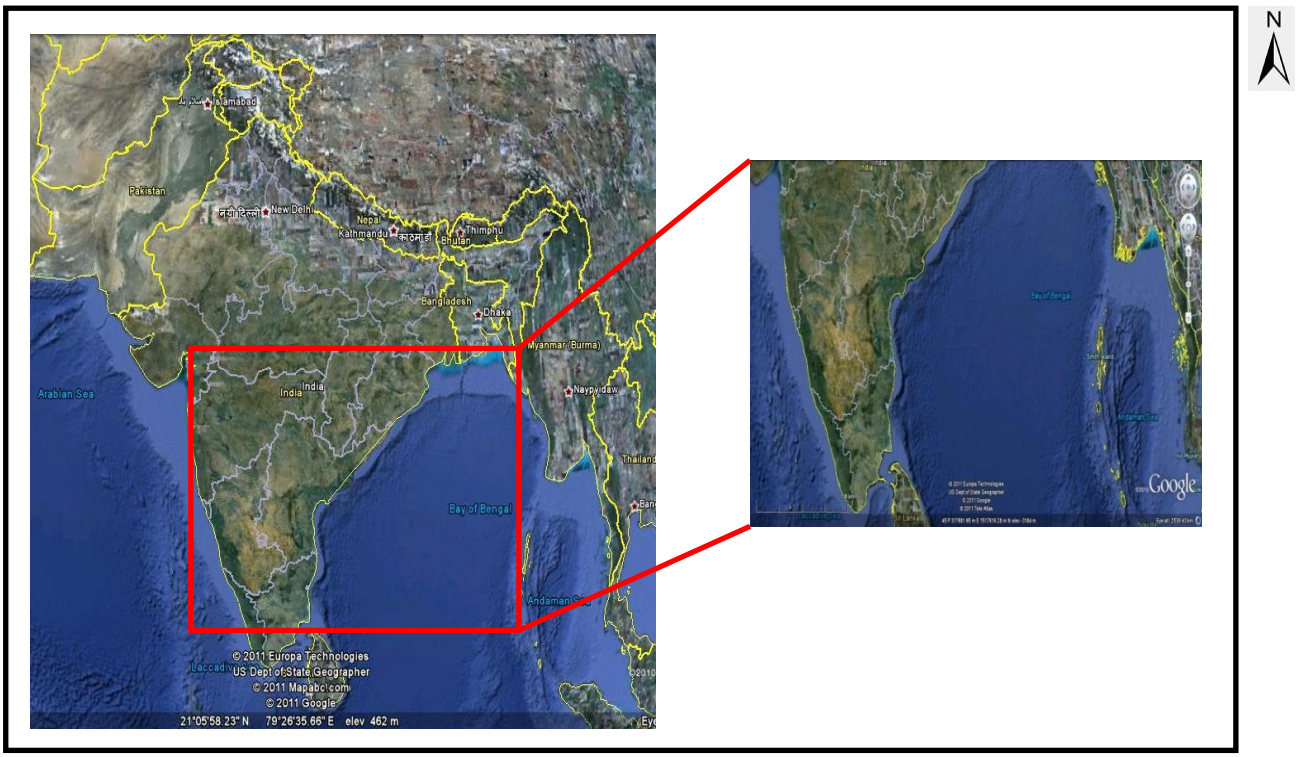


Figure 4-4: Location of area under study (Source: Google Earth accessed on: 20 February, 2011)

The satellite images used in this work are the images of the two cyclones that hit the east coast of India in the year 2009 and 2010. Table 4-2 shows the names of the cyclones and the period of its originating and decaying.

Table 4-2: Cyclone images

Cyclone	Date
Aila	22 May 2009 - 26 May 2009
Laila	15 May 2010 – 19 May 2010

The satellite images are provided by Meteorological and Oceanographic Satellite Data Archival Centre (MOSDAC). The dataset is provided in the hierarchical data format .h5 format. This data was imported in the Integrated GIS software for rescaling and exported to tif format for the further processing in Matlab.

5. RESULTS AND DISCUSSIONS

5.1. Feature extraction

5.1.1. Gray level feature extraction

The gray levels feature vector is represented by the frequencies of pixels that occur in each of the quantized bins and is presented in the Table 5-1.

Table 5-1: Gray level feature vector

Bin Range	Frequencies
0-16	9390
17-32	10854
33-48	10986
49-64	11350
65-80	26959
81-96	18447
97-112	13291
113-128	9720
129-144	8115
145-160	7919
161-176	6820
177-192	6876
193-208	5867
209-224	5047
225-240	5133
241-256	370

5.1.2. Texture feature extraction

The texture feature vector is computed by taking average of all the values of texture descriptors contrast, correlation, homogeneity and energy, computed from the 16 GLCMs. The texture feature vector of the infrared satellite image is presented in the Table 5-2.

Table 5-2: Texture feature vector

	Contrast	Correlation	Homogeneity	Energy
Average value	0.5601	0.9171	0.8228	0.0971

5.1.3. Shape feature extraction:

The shape of the extracted TC from the infrared satellite image is shown in the Figure 5-1.

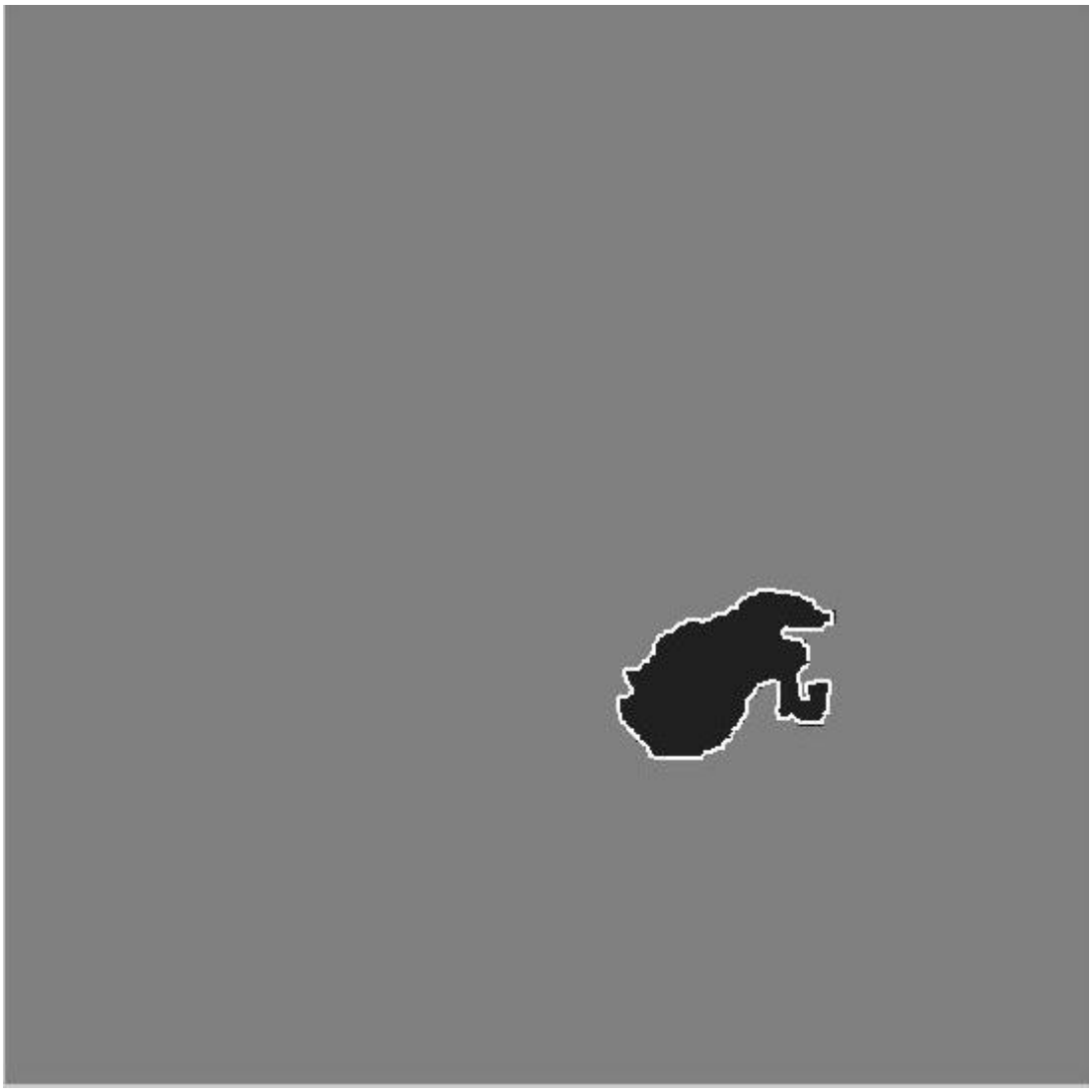


Figure 5-1: Boundary of the extracted T C

The shape feature vector of Figure 5-1 is presented in Table 5-3

Table 5-3: Shape feature vector

	Area	**Perimeter**	**Metric**
Values	3205	342.2082	0.3439

5.2. Database storage and organization

The images and the image feature vectors are stored in the Oracle 10g database. Four empty tables are created in oracle database. First table Images stores the image name and the corresponding images. Second table graylevelfv stores the image name and the corresponding gray level feature vector. Third table texturefv stores image name and texture feature vectors and the fourth table shapefv stores image name and the corresponding shape feature vectors. Image name is the primary key in the table Images and foreign key for the other three tables. Figure 5-2 shows the sample of image database.

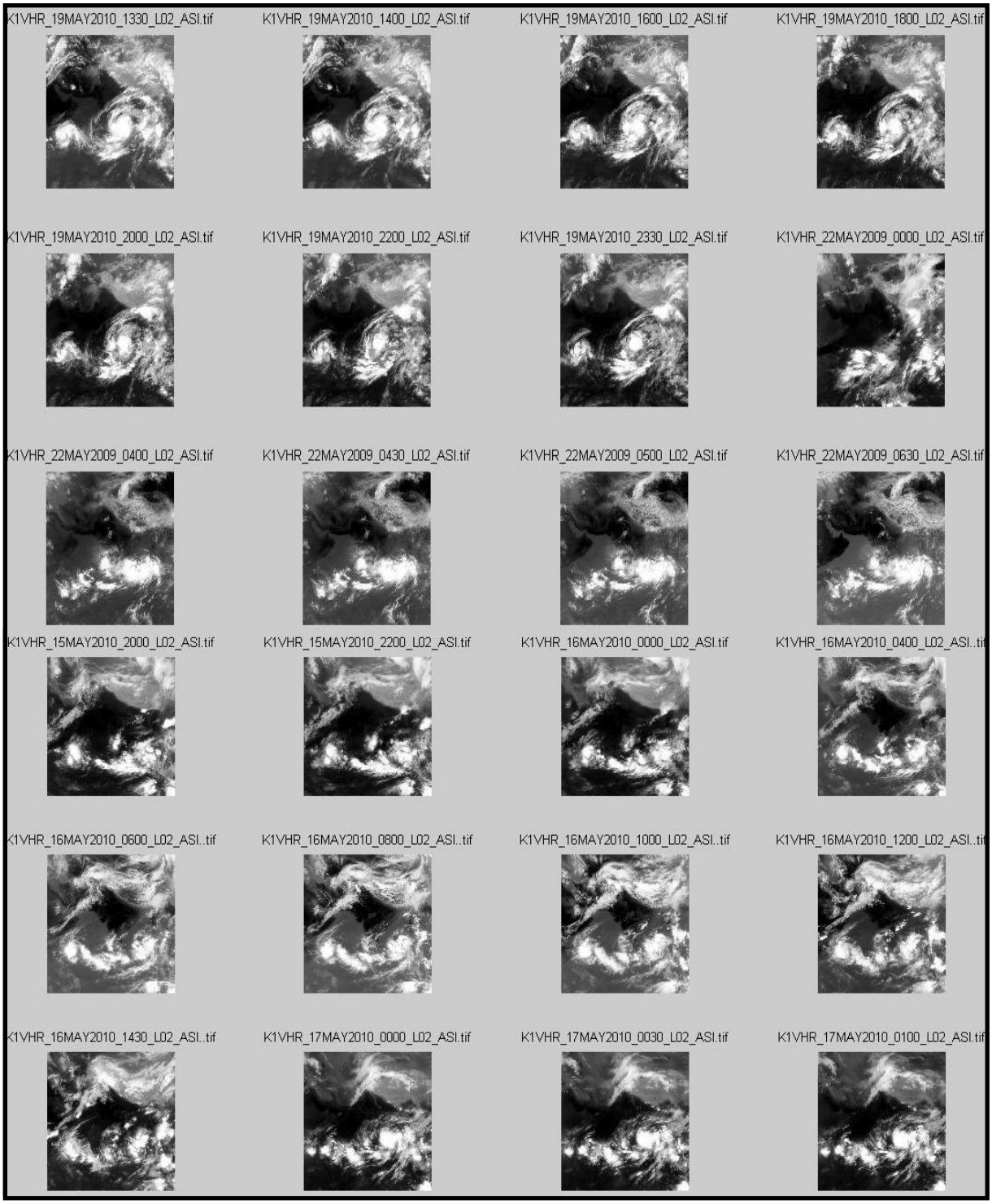


Figure 5-2: Sample image database

5.3. Similarity computation

The similarity between the query image and the images are computed by the Euclidean distance formula. The 12 similarity values for 5 query images are presented from Table 5-4 to Table 5-9.

Table 5-4: Query Image -- K1VHR_18MAY2010_0630_L02_ASI.tif

Image name	Similarity value
'K1VHR_18MAY2010_0630_L02_ASI.tif'	0.0000356
'K1VHR_18MAY2010_0830_L02_ASI.tif'	1570.137149
'K1VHR_17MAY2010_0900_L02_ASI.tif'	2138.436327
'K1VHR_17MAY2010_0730_L02_ASI.tif'	2357.606256
'K1VHR_18MAY2010_1100_L02_ASI.tif'	2368.47267
'K1VHR_15MAY2010_0700_L02_ASI.tif'	2521.456592
'K1VHR_22MAY2009_0600_L02_ASI.tif'	2598.137493
'K1VHR_17MAY2010_0700_L02_ASI.tif'	2937.454588
'K1VHR_22MAY2009_1000_L02_ASI.tif'	3058.969749
'K1VHR_17MAY2010_1030_L02_ASI.tif'	3119.154814
'K1VHR_22MAY2009_0800_L02_ASI.tif'	3340.382456
'K1VHR_17MAY2010_0630_L02_ASI.tif'	3528.365603

Table 5-5: Query Image -- K1VHR_18MAY2010_0830_L02_ASI.tif

Image name	Similarity value
'K1VHR_18MAY2010_0830_L02_ASI.tif'	0.0000205
'K1VHR_18MAY2010_0630_L02_ASI.tif'	3225.930832
'K1VHR_22MAY2009_1000_L02_ASI.tif'	3865.447316
'K1VHR_17MAY2010_0900_L02_ASI.tif'	4677.888211
'K1VHR_22MAY2009_0800_L02_ASI.tif'	5345.271786
'K1VHR_15MAY2010_0700_L02_ASI.tif'	5625.393712
'K1VHR_17MAY2010_0730_L02_ASI.tif'	5717.915149
'K1VHR_18MAY2010_1100_L02_ASI.tif'	5821.114166
'K1VHR_17MAY2010_1030_L02_ASI.tif'	6848.163586
'K1VHR_17MAY2010_0700_L02_ASI.tif'	7329.053664
'K1VHR_22MAY2009_0600_L02_ASI.tif'	7584.72592
'K1VHR_15MAY2010_0930_L02_ASI.tif'	7588.302203

Table 5-6: Query Image -- K1VHR_17MAY2010_1100_L02_ASI.tif

Image name	Similarity value
'K1VHR_17MAY2010_1100_L02_ASI.tif'	0.0000289
'K1VHR_17MAY2010_1030_L02_ASI.tif'	2358.301552
'K1VHR_17MAY2010_0630_L02_ASI.tif'	2821.855806
'K1VHR_15MAY2010_0930_L02_ASI.tif'	3003.277257
'K1VHR_18MAY2010_1100_L02_ASI.tif'	3230.680586
'K1VHR_25MAY2009_0600_L02_ASI.tif'	3449.751886
'K1VHR_17MAY2010_0700_L02_ASI.tif'	3599.408201
'K1VHR_17MAY2010_0730_L02_ASI.tif'	4012.510205
'K1VHR_17MAY2010_1200_L02_ASI.tif'	4243.50806
'K1VHR_15MAY2010_0500_L02_ASI.tif'	4337.827823
'K1VHR_15MAY2010_0700_L02_ASI.tif'	4928.496864
'K1VHR_17MAY2010_0900_L02_ASI.tif'	5111.667366

Table 5-7: Query Image -- K1VHR_15MAY2010_0700_L02_ASI.tif

Image name	Similarity value
'K1VHR_15MAY2010_0700_L02_ASI.tif'	0.00000606
'K1VHR_17MAY2010_0900_L02_ASI.tif'	4018.557349
'K1VHR_18MAY2010_1100_L02_ASI.tif'	4119.005327
'K1VHR_17MAY2010_0730_L02_ASI.tif'	4752.470082
'K1VHR_15MAY2010_0930_L02_ASI.tif'	5094.189933
'K1VHR_17MAY2010_1030_L02_ASI.tif'	5417.767931
'K1VHR_17MAY2010_0700_L02_ASI.tif'	5715.808297
'K1VHR_18MAY2010_0630_L02_ASI.tif'	6953.501793
'K1VHR_18MAY2010_0830_L02_ASI.tif'	7281.920634
'K1VHR_17MAY2010_0630_L02_ASI.tif'	7336.323911
'K1VHR_22MAY2009_0600_L02_ASI.tif'	7804.52184
'K1VHR_17MAY2010_1100_L02_ASI.tif'	9042.765501

Table 5-8: Query Image -- K1VHR_24MAY2009_1600_L02_ASI.tif

Image name	Similarity value
'K1VHR_24MAY2009_1600_L02_ASI.tif'	0.0000351
'K1VHR_24MAY2009_1800_L02_ASI.tif'	2201.495432
'K1VHR_18MAY2010_1500_L02_ASI.tif'	3065.345292
'K1VHR_25MAY2009_0000_L02_ASI.tif'	3095.194487
'K1VHR_24MAY2009_2100_L02_ASI.tif'	3140.468919
'K1VHR_15MAY2010_0000_L02_ASI.tif'	3347.564739
'K1VHR_25MAY2009_0200_L02_ASI.tif'	3480.446034
'K1VHR_15MAY2010_0230_L02_ASI.tif'	3559.522783
'K1VHR_15MAY2010_0100_L02_ASI.tif'	3653.26688
'K1VHR_17MAY2010_1700_L02_ASI.tif'	3779.272512
'K1VHR_17MAY2010_1900_L02_ASI.tif'	4101.045633
'K1VHR_18MAY2010_0000_L02_ASI.tif	4152.944105

5.4. System implementation

Figure 5-3 shows the block diagram of the system operation.

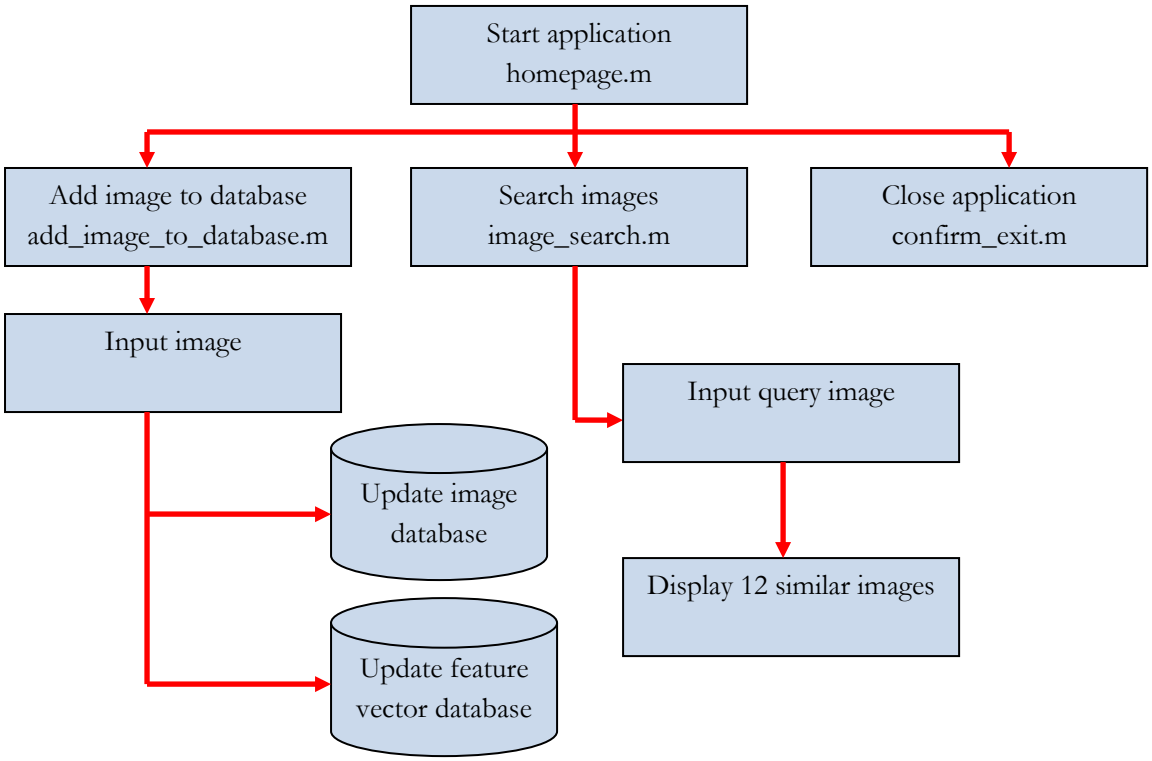


Figure 5-3: Block diagram of the system operation

5.4.1. Start application

By typing homepage on the Matlab command prompt the application is started. Figure 5-4 shows the start of the application. It contains three choices for the user. These choices are add image to database, search for a specific image and closing the application. It performs on the choice of the user. Add image to database option allows user to enter an image and add it to the database with corresponding feature vectors. Search image allows user to search for an image in the database and the close option closes the application on the user confirmation.

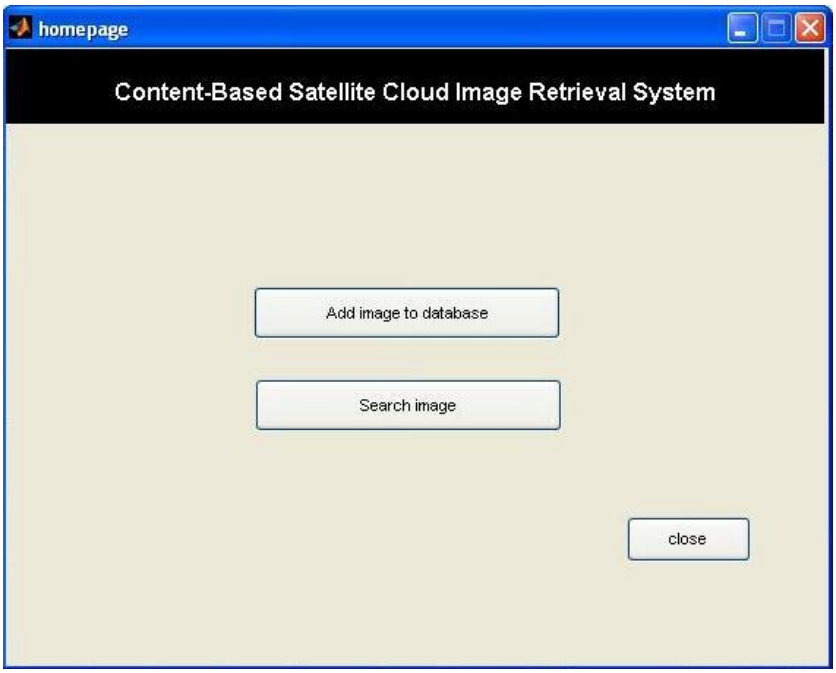


Figure 5-4: Home page of user interface

5.4.2. Add image to database

As can be seen from the Figure 5.4 user has the option of adding image to database. On clicking on the button Add Image to Database the Matlab program add_image_to_database.m is invoked. This lets user to browse for an image and store in the database. Figure 5.5 and Figure 5-6 shows the option to add new image to the database.

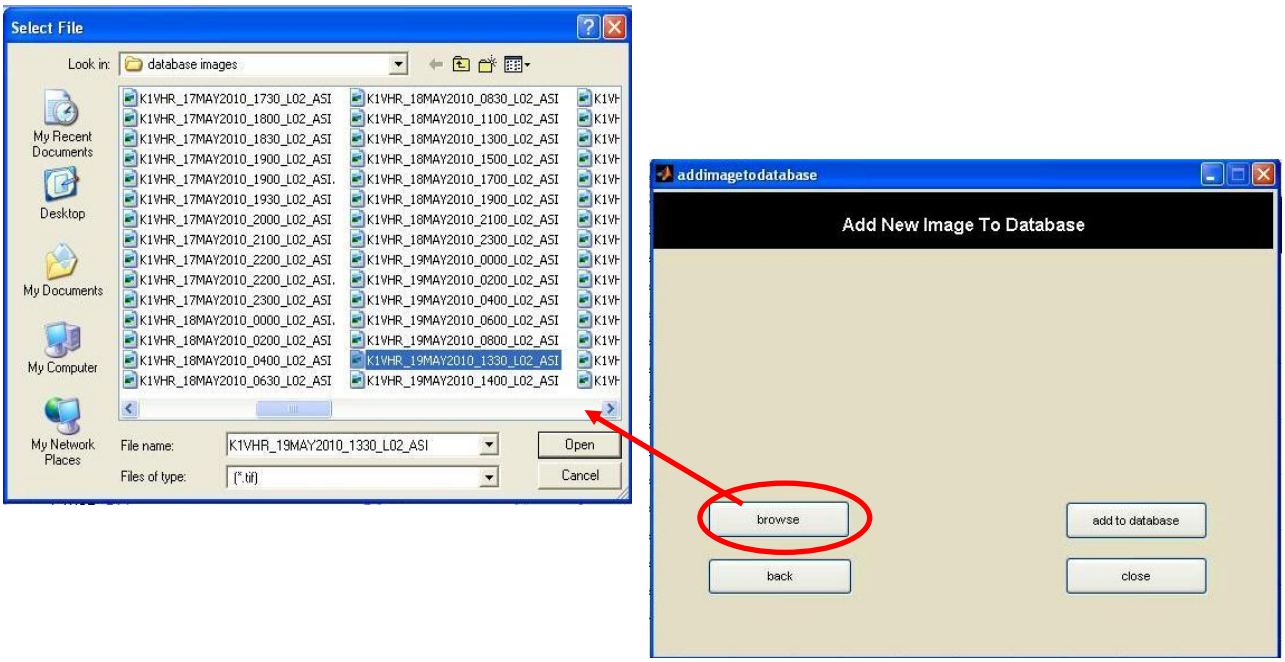


Figure 5-5: Add new image to database

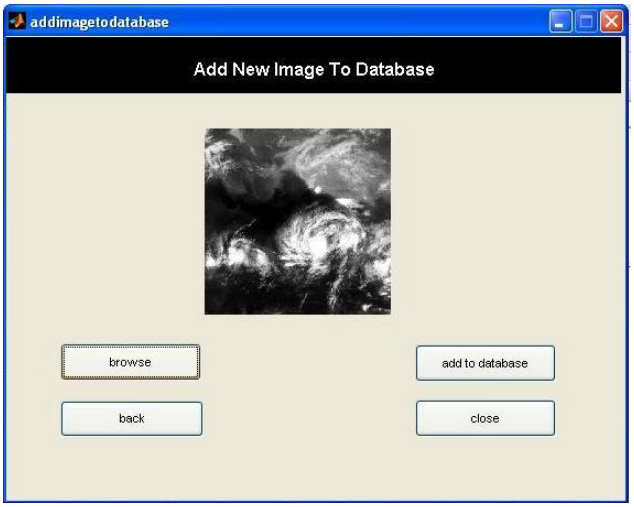


Figure 5-6: Add image to database

On clicking the button add to database the gray levels, texture and shape feature vectors of an image are extracted and stored in the database with the corresponding image.

5.4.3. Searching an image

In the home page of the user interface the option search images allow user to search for a specific image. Figure 5-7 shows the layout of searching an image in the database.

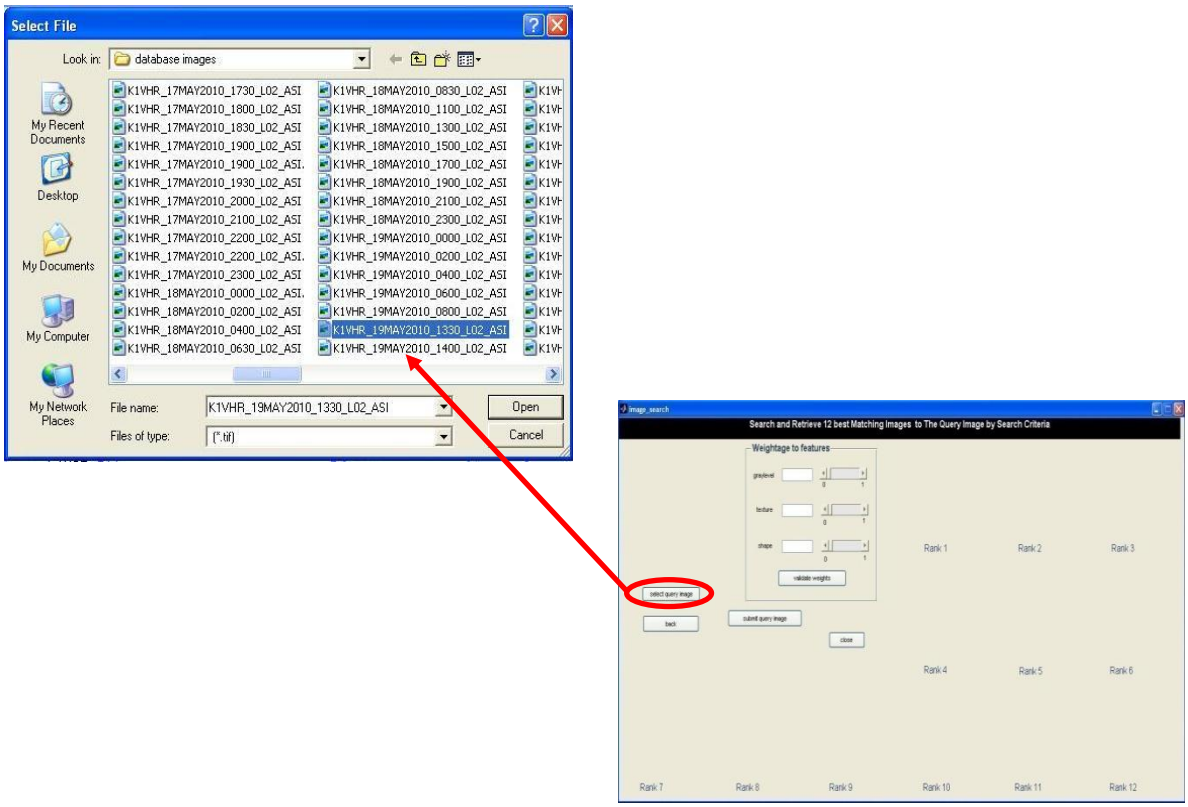


Figure 5-7: Search an image

The select query image button allows user to browse for an image. The user can search images on the basis of the weights assigned to the features. This allows user to search according to the any one of the feature or a combination of the features. Figure 5-8 shows the query image and the weights assigned to the features to search for a specific image.

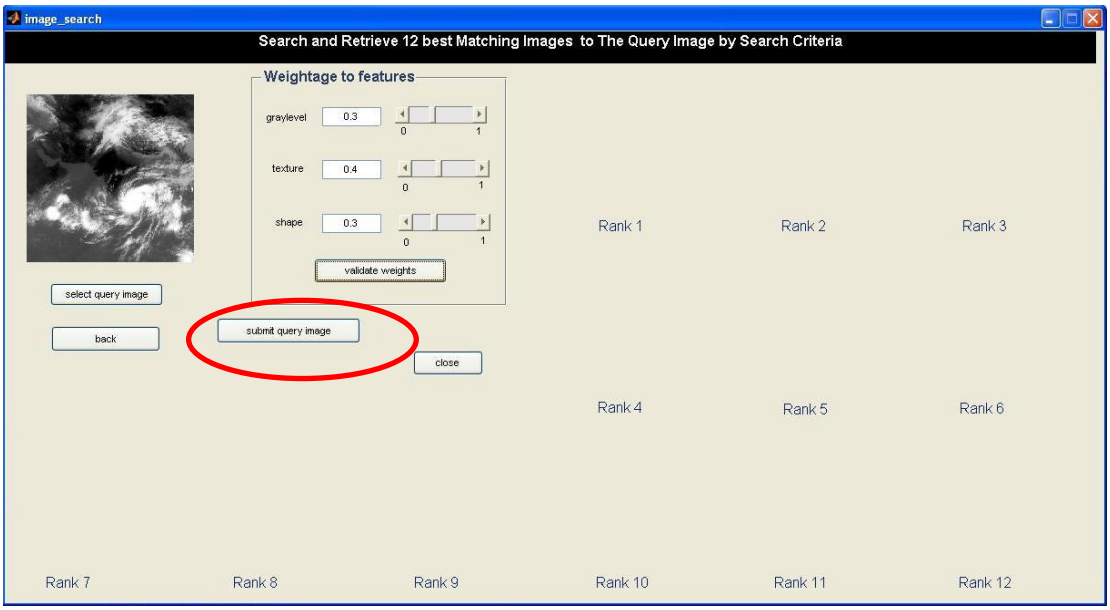


Figure 5-8: (a) Image search

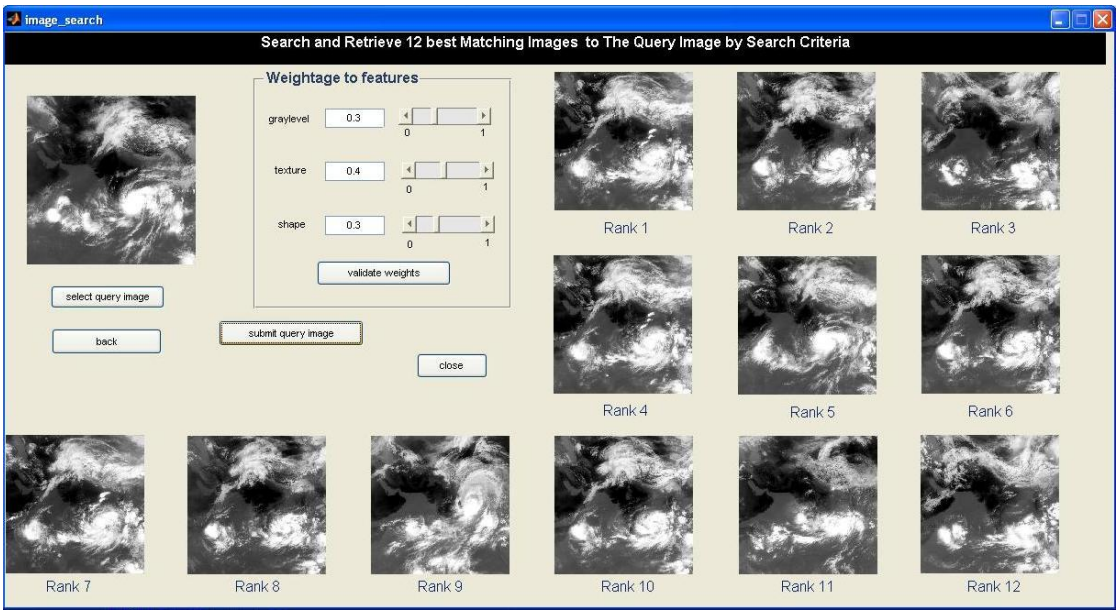


Figure 5-8: (b) Retrieving images by search criteria

5.5. System performance

The performance of the system is evaluated on the basis of the precision of the relevant images retrieved. Precision is the ratio of the number of the relevant images retrieved to the total number of images retrieved.

The image database consists some images of the TC of the year 2009 and 2010. The efficiency of any system cannot be measured with the small image database. As mentioned in [4] precision and recall can measure the efficiency of the system, in this research the performance of the system is checked on the basis of precision.

The different number of images is returned to the user and the precision is calculated for each case. Table 5.9 shows the values of the precision in retrieving the images.

5.5.1. Precision for image retrieval

Table 5-9: Precision values for different methods of image retrieval

Number of retrieved images ↓	Gray level	Texture	Shape	Gray level and texture	Gray level and shape	Texture and shape	All three
5	1	0.4	0.4	1	1	1	1
10	0.5	0.3	0.3	0.5	0.5	0.5	0.6
15	0.33	0.27	0.27	0.33	0.33	0.33	0.4
20	0.25	0.3	0.25	0.25	0.25	0.3	0.3
25	0.2	0.24	0.24	0.2	0.24	0.2	0.24

As can be seen from the Table 5-9 and Figure 5-9 the proposed method of image retrieval performs well for the query image. The precision in retrieving the images is high in the proposed method as compare to the other methods of image retrieval.

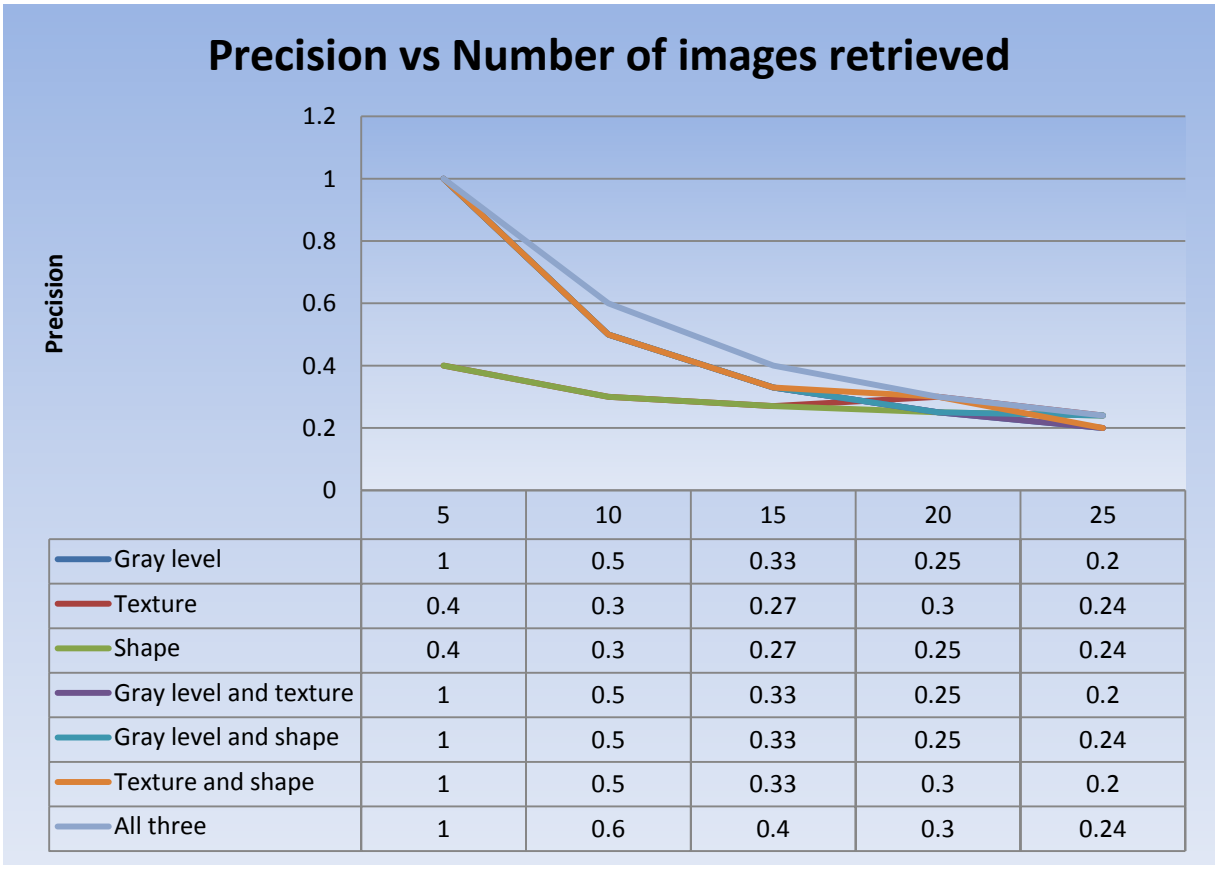
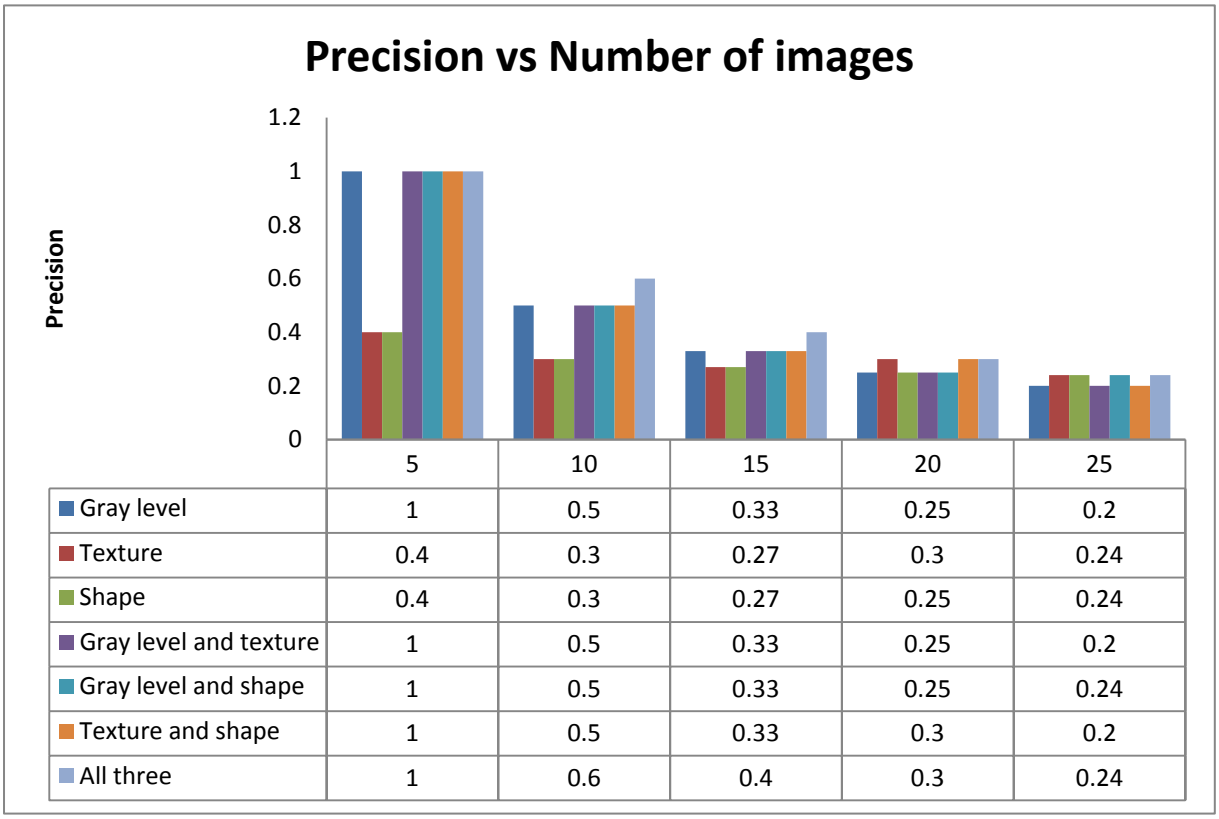


Figure 5-9: Precision vs Number of retrieved images



	5	10	15	20	25
Gray level	1	0.5	0.33	0.25	0.2
Texture	0.4	0.3	0.27	0.3	0.24
Shape	0.4	0.3	0.27	0.25	0.24
Gray level and texture	1	0.5	0.33	0.25	0.2
Gray level and shape	1	0.5	0.33	0.25	0.24
Texture and shape	1	0.5	0.33	0.3	0.2
All three	1	0.6	0.4	0.3	0.24

5-10: Comparison of different approaches of image retrieval

There is not much difference in the retrieval of images based on gray level, texture and shape as Figure 5-10, as compare to other methods. The adopted method of image retrieval in this research is flexible as it provides search on the basis of any of the three features or a combination of them.

5.6. Discussions

The first and the main component of any image retrieval system is the feature extraction. There are many features in the remote sensing image, feature selection is important to make any efficient image retrieval system. Texture is an important feature in the cloud image. Different atmospheric disturbances as typhoons, hurricanes can be extracted by the shape feature and the gray level feature can differentiate between different types of clouds.

By the above mentioned reasons in this research gray level, texture and shape feature are considered as the retrieval features. The algorithms used to extract the gray level feature is the histogram based, feature extraction of texture feature is completed using GLCM approach and the shape feature is extracted using morphological operations.

The histogram approach of image retrieval compare images on the basis of the histogram, the histogram of an image lies between 0 and 255. If the comparison is made on all the discrete points of the histogram, the precision will be high, but proportionally the computational load and time will increase. The histogram based image retrieval needs the optimization of the histogram in the number of bins. In this research different bin widths were considered and their corresponding cost function was computed. The minimum cost was computed at the bin width 16. So, the histogram was split using the bin width equal to 16. The frequencies of the pixels in the bins represent the gray level feature vector.

GLCM method was used to extract the texture feature. The image gray level was rescaled to 0 to 255, to reduce the computational load and time; the gray levels were quantized into 8 × 8 co-occurrence matrix, by selecting the gray levels from 0 to 31 as 1, 32 to 63 as 2 and etc. The satellite cloud image texture is not defined for a particular direction, as the clouds are the dynamic phenomena of the atmosphere. In this research 16 GLCM are created to define the texture of a satellite cloud image. This 16 GLCM are created using the different offsets in the four directions and the four distance vector to the pixel of interest. The four texture measure energy, contrast, correlation and homogeneity are used to define the texture of an image. 16 × 4 = 64 texture measures were computed out of the 16 GLCMs. This is again a large number of comparisons to be made, so the mean of these values were taken to define the texture of an image. Average contrast, Average correlation, Average energy and Average homogeneity were calculated out of these 16 GLCMs. This vector of four values represents the texture feature vector.

Shape feature extraction is completed using the morphological operations. In the first step the noise from the images was removed, by converting into the binary image using an appropriate threshold value. The threshold value that was used in this research is 0.94, based on the experiments on the images used. The noise from the converted binary image was not completely removed and also the boundary of the TC was scattered. To remove the smaller objects from an image and to preserve the shape of a TC, morphological opening was performed. Morphological opening requires the use of the structuring element to perform the opening operation. The choice of the structuring element depends on the shape on which the operation is to be performed. In this research the type of the shape to be extracted is elliptical as TC is characterized by the surrounding spiral shaped cloud patterns. The structuring element used in this research was of the disc type with the size 3. This size was found by the experiments on the images. The area of the TC in the images were more than 1500 pixels, after the morphological opening operation the objects containing less than 1500 pixels were removed from the image. The finally extracted shape of a TC still contains gaps. To fill the gaps inside the object the morphological filling operation was used. After the morphological filling operation the shape of a TC was extracted. Using the region properties, perimeter, area and metric were calculated to make a shape feature vector.

The images and the feature vectors are stored in the Oracle database 10g. Oracle database is having the storage of peta bytes of the data and also provides functionalities to manage raster data. Four empty tables in the Oracle database were created to store the images and the feature vectors. The images and the extracted feature vectors are added to the database through user interface. At any time the system allows the user to search for an image in the database. The image database consists of the 45 images.

User interface is created using the Matlab GUIDE tool. This allows the user to enter the query image and search for the similar images in the database.

Similarity between the query image and the images in the database is computed using the Euclidean distance formula. The similarity between the gray level feature vectors, texture feature vectors and the shape feature vectors are computed separately. These values of similarity for each image are added to compute the similarity of any image in the database to the query image. User interface also allows the user to search for an image based on any feature or a combination of them, by assigning weights to the different features. This provides more flexibility to search images. The final similarity value is calculated by multiplying the gray level weight age to the gray level similarity, texture weight age to the texture similarity and the shape weight age to the shape similarity and adding them.

These similarity values is sorted by using sorting algorithm, the top 12 most similar images to the query image are ranked and presented to the user.

The satellite images used in this work contains images of cyclones in the year 2009 and 2010. The performance of the system is evaluated on the basis of the precision in retrieving the images. Different numbers of images are retrieved from the image database and the precision is calculated as from the formula mentioned in equation (14).The precision is high in case of retrieving the images on the basis of the three features gray level, texture and shape.

6. CONCLUSIONS AND RECOMMENDATIONS

6.1. Conclusions

6.1.1. What methods will be used to extract the features?

The satellite cloud image is a valuable source that contains information about the atmosphere. There are many features in the remote sensing image. The features used in this research are gray level, texture and shape of a TC. The method used to extract the gray level feature is the histogram approach, due to its efficiency and easy computation.

The GLCM method is used to extract the texture feature vector; it is the statistical method to compute texture. In [34] 14 measures are defined that describe the texture of an image. The GLCM method of texture based image retrieval considers the spatial relationships of pixels. It allows computing the texture of an image by specifying the offsets and the distance vector, this offsets can be defined in different directions with different distances with the pixel of interest.

In literature different methods of shape based image retrieval are defined, the most common method of extracting the shape of a TC is the gradient vector flow (GVF) snake model. The disadvantages of these methods are many point to point calculations while comparing the similar images of TCs. In this research image retrieval is based on gray level, texture and shape feature, to reduce the complexity and computational time the morphological operations are used to extract the shape feature.

6.1.2. What will be the appropriate way of storing the images and the feature vectors?

The images and the feature vectors can be stored in the database or file system. Database offers several advantages over file system. The retrieval from database is comparatively faster than the file system, the other advantages of the databases are security and redundancy can be avoided in the databases. Oracle database after the release of Oracle 10g, also offers many inbuilt functionalities to store and manage raster data. Some of the functionalities of the Oracle Spatial are geo-referencing, image tiling, pyramiding, and indexing. The speed of the retrieval can be improved using the indexing technique; Oracle Spatial 10g provides R-tree indexing.

On the other hand, file system is slower than the database. Due, to the advantages of the database over the file system, the images and the extracted feature vectors are stored in the Oracle database 10g.

6.1.3. What will be the method of calculating the similarity between the images?

The different methods of calculating the similarity between the images are the Euclidean distance, Minkowski distance, Quadratic distance and etc. The most common method of calculating the similarity between the images is the Euclidean distance. In this research the Euclidean distance is used to find the similarity between the images. The similarity between the gray level, texture and shape feature are computed separately, to allow the user to provide the weight age to any feature. After the assignment of the weights to the different features final similarity value is computed between the images.

6.1.4. How can the images be ranked and retrieved?

The images are ranked according to the similarity value. Using the sorting algorithm the images similarity values are sorted and arranged in the ascending order of the similarity value. The images corresponding to the first 12 similarity values are retrieved and displayed to the user.

6.1.5. What will be the method to evaluate the performance of the system?

The performance of the system is evaluated on the basis of the precision in retrieving the images. The precision is calculated as the ratio of the number of relevant images to the total number of images retrieved. The relevant images to the query image that are considered in this research are the images of the same day to the query image.

6.2. Recommendations

Satellite cloud contains valuable information that can be used in different applications. Due to the huge quantity of the image data, the storage and the management of the images are necessary. The creation of satellite image database requires efficient tools for searching and browsing the image database. Text based image retrieval is not efficient for the retrieval of images, the CBIR approach is more effective as compare to the text based image retrieval as it uses the contents of images rather than the textual annotations. The CBIR approach of image retrieval is based on different components these are feature extraction, database storage, similarity computations and user interface.

Feature extraction is an important part in CBIR approach, there are many features in the remote sensing image, in this research three features gray level, texture and shape are used to retrieve images. The accuracy of the image retrieval can be improved by using the other features of the satellite cloud image as spatial relation, acreage.

To speed up the retrieval of the images some indexing method can be used. In this research the indexing scheme is not used. Oracle Spatial 10g provides R-tree indexing. The way of storing the images in the database can be changed, the images can be stored in the form of image tiles, further the pyramiding and other functionalities of the Oracle database can be used to improve the system efficiency.

The performance of the system can be better evaluated by the standard precision and recall graphs, with the increasing number of images. This system can be further improved and used as a tool to compare the images in real time, for other applications as forest fires, dust storms and etc.

LIST OF REFERENCES

[1] Anonymous, *Disaster Management in India*, Indian Meteorological Department, .

[2] P. Blanchart and M. Datcu, "A Semi-Supervised Algorithm for Auto-Annotation and Unknown Structures Discovery in Satellite Image Databases," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, vol. 3, 2010, pp. 698-717.

[3] Y. Hao, W. ShangGuan, Y. Zhu, and Y. Tang, "Contented-Based Satellite Cloud Image Processing and Information Retrieval," *Bio-Inspired Computational Intelligence and Applications*, Springer Berlin / Heidelberg, 2007, pp. 767-776-776.

[4] F. Dell'Acqua and P. Gamba, "Query-by-shape in meteorological image archives using the point diffusion technique," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 39, 2001, pp. 1834-1843.

[5] Wei ShangGuan, YanLing Hao, YanHong Tang, and Yi Zhu, "The Research and Application of Content-Based Satellite Cloud Image Retrieval," *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, 2007, pp. 3864-3869.

[6] E. Jones and A. Roydhouse, "Intelligent retrieval of archived meteorological data," *IEEE Expert*, vol. 10, 1995, pp. 50-58.

[7] R. Holowczak, F. Artigas, Soon Ae Chun, June-Suh Cho, and H. Stone, "An experimental study on content-based image classification for satellite image databases," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 40, 2002, pp. 1338-1347.

[8] Chi-Ren Shyu, M. Klaric, G. Scott, A. Barb, C. Davis, and K. Palaniappan, "GeoIRIS: Geospatial Information Retrieval and Indexing System—Content Mining, Semantics Modeling, and Complex Queries," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 45, 2007, pp. 839-852.

[9] M. Datcu, H. Daschiel, A. Pelizzari, M. Quartulli, A. Galoppo, A. Colapicchioni, M. Pastori, K. Seidel, P. Marchetti, and S. D'Elia, "Information mining in remote sensing image archives: system concepts," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 41, 2003, pp. 2923-2936.

[10] Y. Alemu, Jong-bin Koh, M. Ikram, and Dong-Kyoo Kim, "Image Retrieval in Multimedia Databases: A Survey," *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on*, 2009, pp. 681-689.

[11] S. Subramanya, Jui-Che Teng, and Yongjian Fu, "Study of relative effectiveness of features in content-based image retrieval," *Cyber Worlds, 2002. Proceedings. First International Symposium on*, 2002, pp. 168-175.

[12] M.E. Osadebey, "Integrated Content-based image retrieval using texture, shape and spatial information," Umea University, Umea, 2006.

[13] P. Carrara, G. Pasi, M. Pepe, and A. Rampini, "An Indexing Model of Remote Sensing Images," *Image and Video Retrieval*, Springer Berlin / Heidelberg, 2004, pp. 240-240.

[14] S. Sakji-Nsibi and A. Benazza-Benyahia, "Multispectral image indexing based on Vector Lifting Schemes," *Geoscience and Remote Sensing Symposium,2009 IEEE International,IGARSS 2009*, 2009, pp. IV-482-IV-485.

[15] G. Camps-Valls and L. Bruzzone, "Kernel Multivariate Analysis in Remote Sensing Feature Extraction," *Kernel Methods for Remote Sensing Data Analysis*, John Wiley & Sons, 2009, pp. 329-352.

[16] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, 2000, pp. 1349-1380.

[17] L.L. Janssen and B.G. Gorte, “Digital Image Classification,” *Principles of Remote Sensing*, The International Institute for Geo-Information Science and Earth Observation, 2004, pp. 193-204.
[18] X. Wang, Y. Yu, and H. Yang, “An effective image retrieval scheme using color, texture and shape features,” *Computer Standards & Interfaces*, vol. 33, Jan. 2011, pp. 59-68.
[19] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Qian Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, “Query by image and video content: the QBIC system,” *Computer*, vol. 28, 1995, pp. 23-32.
[20] W. Ma and B. Manjunath, “NeTra: a toolbox for navigating large image databases,” *Image Processing, 1997. Proceedings., International Conference on*, 1997, pp. 568-571 vol.1.
[21] S. Omar, M. Ismail, and S. Ghanem, “WAY-LOOK4: A CBIR system based on class signature of the images' color and texture features,” *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, 2009, pp. 464-471.
[22] J.R. Smith and S. Chang, “VisualSEEk: a fully automated content-based image query system,” *Proceedings of the fourth ACM international conference on Multimedia*, Boston, Massachusetts, United States: ACM, 1996, pp. 87-98.
[23] H.D. Benitez and G.I. Alvarez, “Content Based Thermal Images Retrieval,” *Image and Signal Processing*, 2010, pp. 479-487.
[24] Janghyun Yoon and N. Jayant, “Relevance feedback for semantics based image retrieval,” *Image Processing, 2001. Proceedings. 2001 International Conference on*, 2001, pp. 42-45 vol.1.
[25] Hu Min and Yang Shuangyuan, “Overview of content-based image retrieval with high-level semantics,” *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, 2010, pp. V6-312-V6-316.
[26] S. Monir and S. Hasnain, “A Framework for Interactive Content-Based Image Retrieval,” *9th International Multitopic Conference, IEEE INMIC 2005*, 2005, pp. 1-4.
[27] Janghyun Yoon and N. Jayant, “Relevance feedback for semantics based image retrieval,” *Image Processing, 2001. Proceedings. 2001 International Conference on*, 2001, pp. 42-45 vol.1.
[28] Xin Zhang, Bing Wang, Zhi-De Zhang, and Xiao-Yan Zhao, “SSVR-based image semantic retrieval,” *Machine Learning and Cybernetics, 2008 International Conference on*, 2008, pp. 2607-2611.
[29] I. Dimitrovski, P. Guguljanov, and S. Loskovska, “Implementation of web-based medical image retrieval system in Oracle,” *Adaptive Science & Technology, 2009. ICAST 2009. 2nd International Conference on*, 2009, pp. 192-197.
[30] Seung-Hoon Lee, Gi-Hwa Jang, Su-Hyun Lee, Sung-Hwan Jung, and Yong-Tae Woo, “A content-based image retrieval system using extended SQL in RDBMS,” *Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on*, 1997, pp. 1069-1072 vol.2.
[31] T. Jaworska, “The Inner Structure of Database for the CBIR System,” *Computational Intelligence for Modelling Control & Automation, 2008 International Conference on*, 2008, pp. 13-18.
[32] L. Chiorean, A. Sipos, and M. Vaida, “Radiology Database Images Retrieving,” *Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on*, 2007, pp. 1-4.
[33] P. Maheshwary and N. Sricastava, “Prototype System for Retrieval of Remote Sensing Images based
on Color Moment and Gray Level Co-Occurrence Matrix,” *International Journal of Computer Science Issues,*, vol. 3, 2009.
[34] R.M. Haralick, K. Shanmugam, and I. Dinstein, “Textural Features for Image

Classification," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 3, 1973, pp. 610-621.
[35] A. Ma and I. Sethi, "Local shape association based retrieval of infrared satellite images," *Multimedia, Seventh IEEE International Symposium on*, 2005, p. 7 pp.
[36] J. Liu, Bo Feng, Meng Wang, and Weidong Luo, "Tropical Cyclone Forecast using Angle Features and Time Warping," *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, 2006, pp. 4330-4337.
[37] Chenyang Xu and J. Prince, "Gradient vector flow: a new external force for snakes," *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, 1997, pp. 66-71.
[38] Chenyang Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *Image Processing, IEEE Transactions on*, vol. 7, 1998, pp. 359-369.
[39] Fengbo Wu, Yuan-Xiang Li, Peng Xu, and Xudong Liang, "Image Retrieval Using Ellipse Shape Feature with Particle Swarm Optimization," *Multimedia Technology (ICMT), 2010 International Conference on*, 2010, pp. 1-4.
[40] G. Awcock and R. Thomas, "Image Preprocessing," *Applied Image Processing*, The Macmillan press Ltd., 1996, pp. 90-125.
[41] R. Chakravarti and Xiannong Meng, "A Study of Color Histogram Based Image Retrieval," *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, 2009, pp. 1323-1328.
[42] N. Sai and R. Patil, "New Feature Vector for Image Retrieval: Sum of Value of Histogram Bins," *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT '09. International Conference on*, 2009, pp. 550-554.
[43] A.K. Jain, "Image Retrieval using Color and Shape," *Pattern Recognition*, vol. 29, 1996, pp. 1233-1244.
[44] H. Shimazaki and S. Shinomoto, "A Method for Selecting the Bin Size of a Time Histogram," *Neural Comput.*, vol. 19, Jun. 2007, pp. 1503-1527.
[45] R. Gonzalez, *Digital image processing*, Dorling Kindersley ;;Pearson Prentice Hall, 2009.
[46] D. Clausi, "An Analysis of Co-occurrence Texture Statistics as a Function of Grey Level Quantization," *Canadian Journal of Remote Sensing*, vol. 28, 2002, pp. 45-62.
[47] Anonymous, "Specifying the Offsets, Matlab Help File."
[48] I. Mocanu, "Image Retrieval by Shape Based on Contour Techniques A Comparative Study," *Applied Computational Intelligence and Informatics, 2007. SACI '07. 4th International Symposium on*, 2007, pp. 219-223.
[49] R. Lee and J. Lin, "An elastic contour matching model for tropical cyclone pattern recognition," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 31, 2001, pp. 413-417.
[50] J. You and P. Bhattacharya, "Dynamic shape retrieval by hierarchical curve matching, snakes and data mining," *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 2000, pp. 1035-1038 vol.1.
[51] Q. Zhang, L. Lai, and W. Sun, "Location of Tropical Cyclone Center with Intelligent Image Processing Technique," *Advances in Machine Learning and Cybernetics*, 2006, pp. 898-907.
[52] Anonymous, "Morphological Operations," *Image Processing Toolbox TM 7 User's Guide*, The MathWorks, Inc., , pp. 10-1 - 10-46.
[53] Zhihua Xie, Guodong Liu, Cuiqun He, and Yujie Wen, "Texture Image Retrieval Based on Gray Level Co-Occurrence Matrix and Singular Value Decomposition," *Multimedia Technology (ICMT), 2010 International Conference on*, 2010, pp. 1-3.
[54] Zhang Rui-zhe, Yuan Jia-zheng, Huang Jing-hua, Wang Yu-jian, and Bao Hong, "A

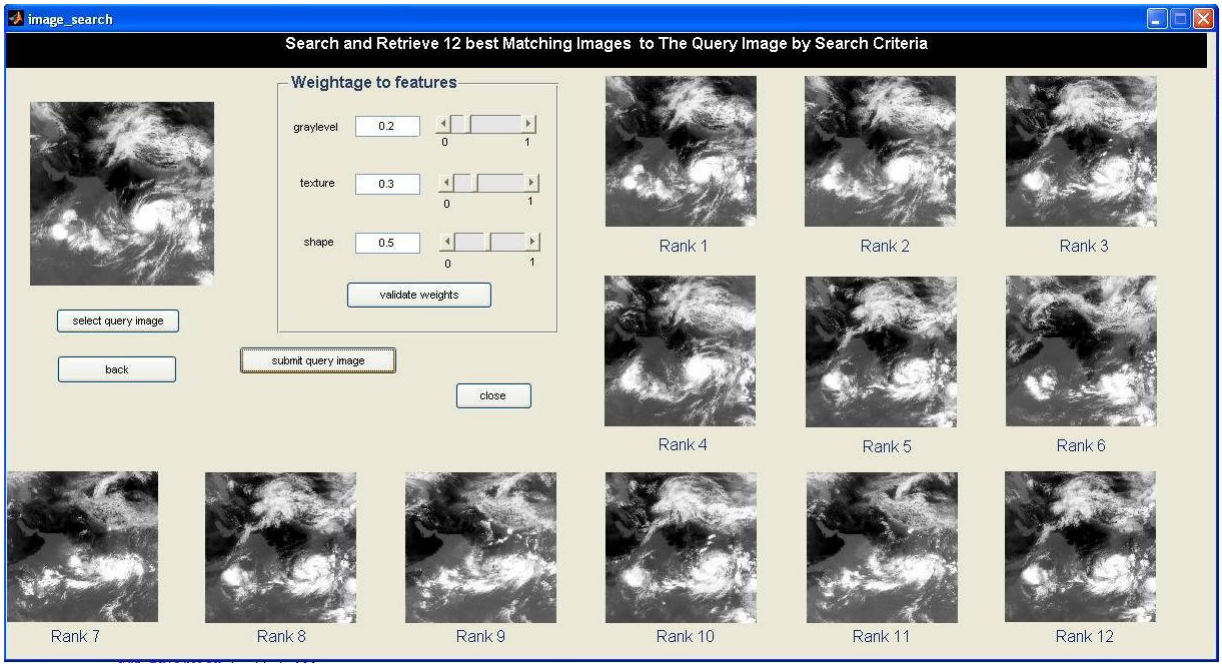
Novel Generalized SVM Algorithm with Application to Region-Based Image Retrieval," *Information Technology and Applications, 2009. IFITA '09. International Forum on*, 2009, pp. 280-283.

APPENDIX A: GLOSSARY

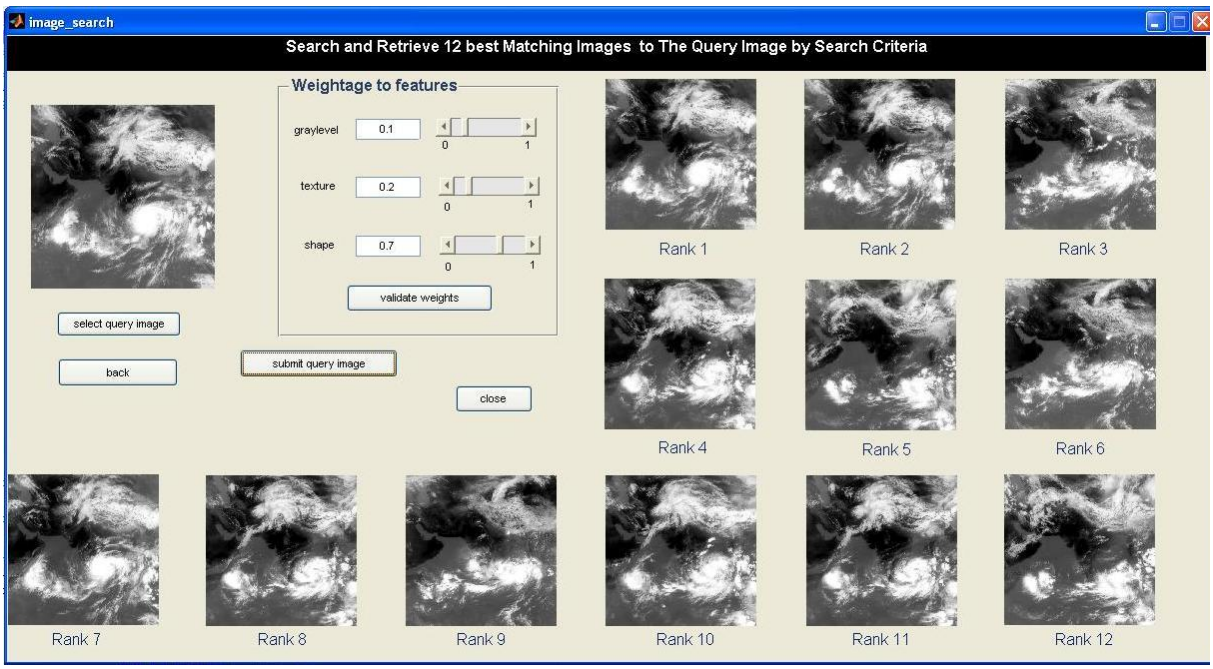
TC	Tropical Cyclone, A naturally occurring, destructive phenomena in tropics.
CBIR	Content-Based Image Retrieval, An approach of image retrieval based on the contents of the images
Feature	Any distinguishable characteristic in an image
visual features in CBIR	Colour, texture and shape
Feature extraction	The method of extracting the features from an image, which is used to reduce the dimensions of image data
Quantization	Reducing the number of bins in a optimal manner so that gray levels very similar stays in a single bin
Feature vector	A vector, set of values that represent a feature in an image
Similarity	It is the distance between image feature vectors
Semantics	High level image features, close to human perception, difficult to implement
Signatures	In remote sensing image, each surface type is represented by a set of digital numbers, in each spectral band, called the image signatures

Appendix B: Image Retrieval using adopted method

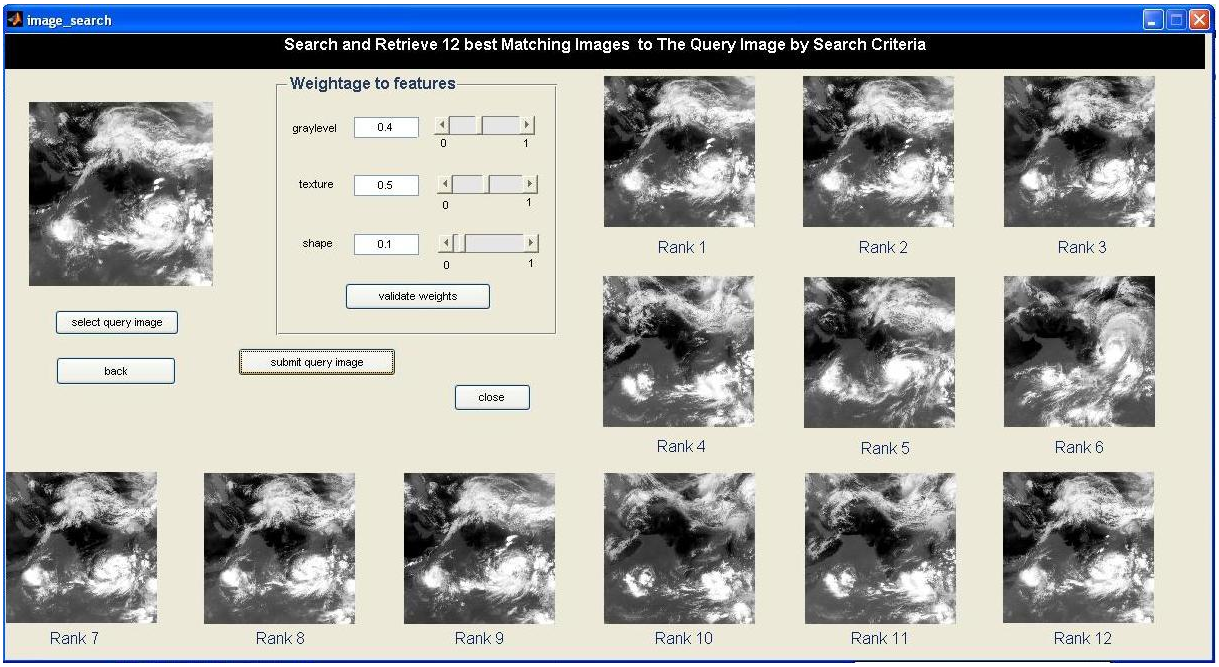
Query Image1:



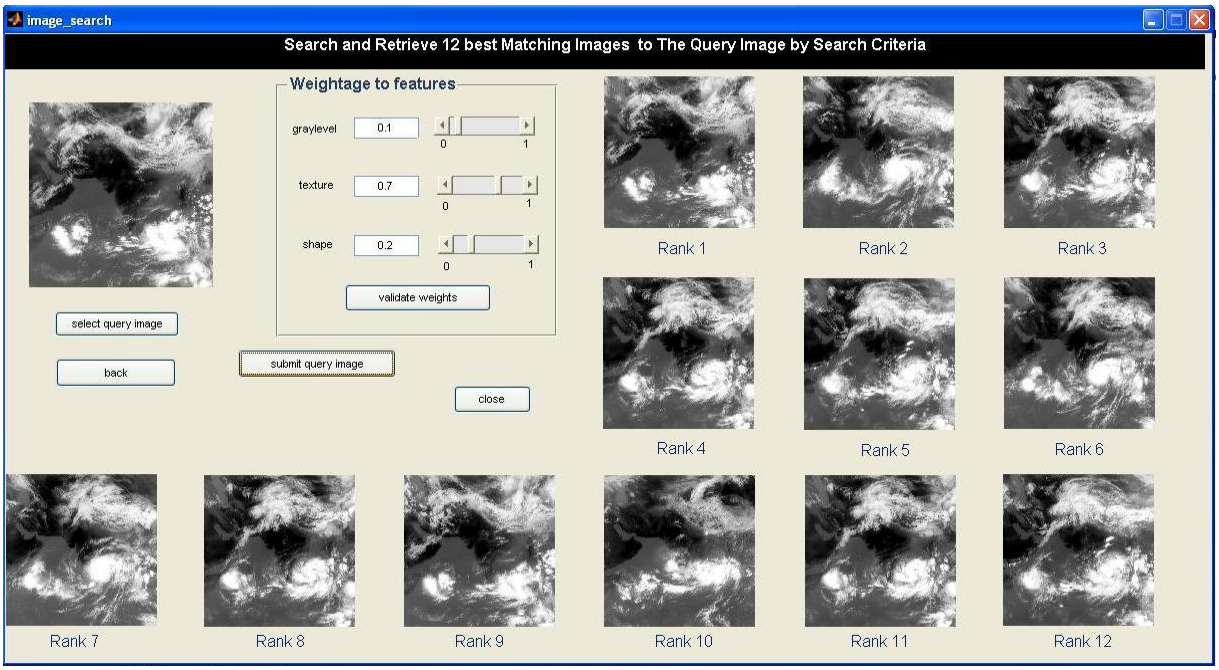
Query Image2:



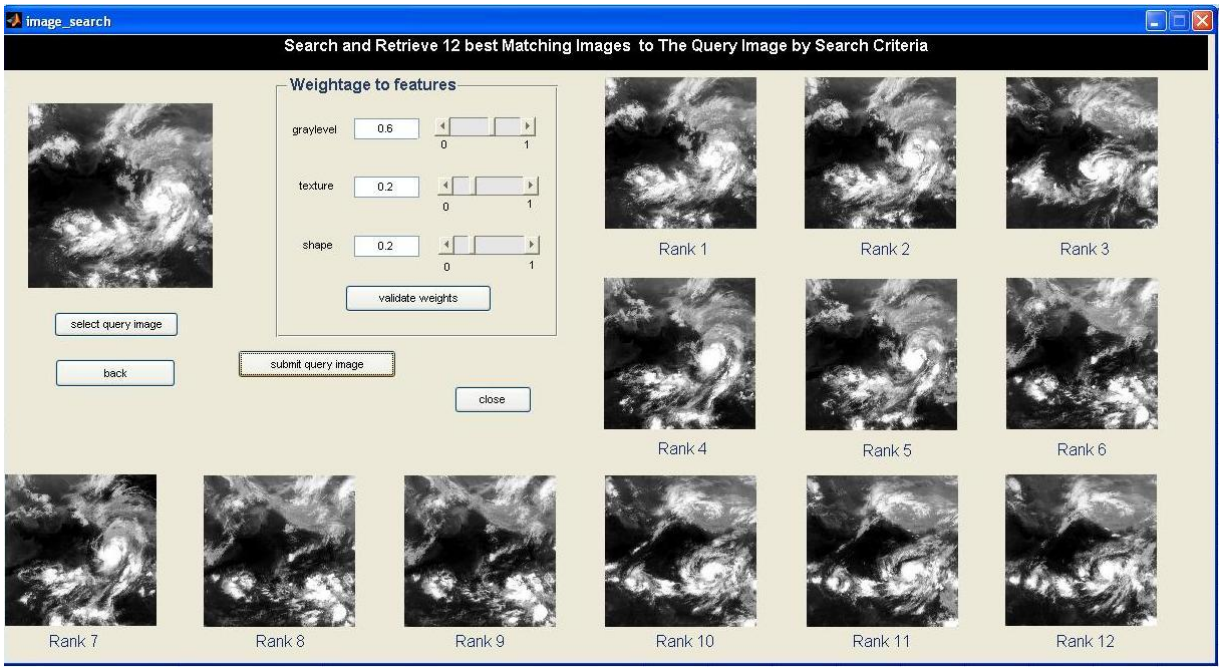
Query Image3:



Query Image4:



Query Image5:



APPENDIX C: MATLAB CODES

homepage.m (Home page of user interface)

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

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

% Last Modified by GUIDE v2.5 08-Mar-2011 14:57:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @homepage_OpeningFcn, ...
                   'gui_OutputFcn',  @homepage_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before homepage is made visible.
function homepage_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 homepage (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes homepage wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = homepage_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;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

addimagetodatabase;

% --- Executes on button press in search.
function search_Callback(hObject, eventdata, handles)
% hObject    handle to search (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

image_search;

% --- Executes on button press in close1.
function close1_Callback(hObject, eventdata, handles)
% hObject    handle to close1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

pos_size = get(handles.figure1,'Position');

user_response = confirm_exit('Title','Confirm close');
switch user_response
    case {'No'}
        % do nothing
    case 'Yes'
        delete(handles.figure1);

end
```

add_to_database.m (Adding new image to Database)

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

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

% Last Modified by GUIDE v2.5 08-Mar-2011 15:08:08

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @addimagetodatabase_OpeningFcn, ...
                   'gui_OutputFcn',  @addimagetodatabase_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before addimagetodatabase is made visible.
```

```
function addimagetodatabase_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 addimagetodatabase (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes addimagetodatabase wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = addimagetodatabase_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;


% --- Executes on button press in Browse.
function Browse_Callback(hObject, eventdata, handles)
% hObject    handle to Browse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[FileName, folder] = uigetfile({'*.tif'}, 'Select File');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end

J = imread(fullName);
imshow(J,'Parent',handles.axes1);
set(handles.Browse, 'userdata', fullName)
set(handles.Browse, 'userdata', FileName)
guidata(hObject,handles);



% --- Executes on button press in Back.
function Back_Callback(hObject, eventdata, handles)
% hObject    handle to Back (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

homepage;

% --- Executes on button press in add.
```

```
function add_Callback(hObject, eventdata, handles)
% hObject    handle to add (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

FileName = get(handles.Browse,'userdata');
fullName = get(handles.Browse,'userdata');

img = imread(fullName);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%  CALCULATING GRAY LEVEL FEATURE VECTORS OF THE LOADED IMAGE %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Histogram of an image

x_axis = 0:16:255;
y_axis = img(1:end);
% hist(y_axis,x_axis);

frequency = histc(y_axis,x_axis);
frequency(1);
frequency(2);
frequency(3);
frequency(4);
frequency(5);
frequency(6);
frequency(7);
frequency(8);
frequency(9);
frequency(10);
frequency(11);
frequency(12);
frequency(13);
frequency(14);
frequency(15);
frequency(16);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%  CALCULATING TEXTURE FEATURE VECTORS OF INPUT IMAGE  %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

GLCMS = graycomatrix(img,'Offset',[0 1; 0 2; 0 3; 0 4;-1 1; -2 2;...
    -3 3; -4 4;
-1 0; -2 0; -3 0; -4 0;-1 -1; -2 -2; -3 -3; -4 -4]);

contrast = graycoprops(GLCMS,'contrast');
correlation = graycoprops(GLCMS,'correlation');
energy = graycoprops(GLCMS,'energy');
homogeneity = graycoprops(GLCMS,'homogeneity');

% Calculating mean of all the contrast of 16 GLCMS

sum_contrast = 0;
for i = 1:16
    sum_contrast = sum_contrast + contrast.Contrast(i);
    mean_contrast = sum_contrast/16;
```

```
end
mean_contrast;

% Calculating mean of all the correlation of 16 GLCMS

sum_correlation = 0;
for i = 1:16
    sum_correlation = sum_correlation + correlation.Correlation(i);
    mean_correlation = sum_correlation/16;
end
mean_correlation;

% Calculating mean of all the homogeneity of 16 GLCMS

sum_homogeneity = 0;
for i = 1:16
    sum_homogeneity = sum_homogeneity + homogeneity.Homogeneity(i);
    mean_homogeneity = sum_homogeneity/16;
end
mean_homogeneity;

% Calculating mean of all the energy of 16 GLCMS

sum_energy = 0;
for i = 1:16
    sum_energy = sum_energy + energy.Energy(i);
    mean_energy = sum_energy/16;
end
mean_energy;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  CALCULATING SHAPE FEATURE VECTORS OF INPUT IMAGE  %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Convert input image into binary image using threshold value

binary_image = im2bw(img,0.94);
% figure, imshow(binary_image);

% Choosing the structuring element

se = strel('disk',3);

% Morphological operation for image smoothing

openedBW = imopen(binary_image,se);
% figure, imshow(openedBW);



bw = bwareaopen(openedBW,1500);
% figure, imshow(bw)

% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
bw = imfill(bw,'holes');
```

```
% figure,imshow(bw)


[B,L] = bwboundaries(bw,'noholes');

% figure,imshow(bw)

stats = regionprops(L,'Area','Centroid');

threshold = 0.94;

% loop over the boundaries
for k = 1:length(B)

  % obtain (X,Y) boundary coordinates corresponding to label 'k'
  boundary = B{k};

  % compute a simple estimate of the object's perimeter
  delta_sq = diff(boundary).^2;
  perimeter = sum(sqrt(sum(delta_sq,2)));

  % obtain the area calculation corresponding to label 'k'
  area = stats(k).Area;

  % compute the roundness metric
  metric = 4*pi*area/perimeter^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%  CONNECTING TO THE ORACLE DATABASE  %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

img = reshape(img,[],1);
javaaddpath 'D:/oracle/product/10.2.0/db_1/jdbc/lib/ojdbc14.jar';
conn = database('oracle','system','deepak',...
   'oracle.jdbc.driver.OracleDriver', ....
   'jdbc:oracle:thin:');

a = isconnection(conn);
% ping(conn);
if a == 0;
    disp ('connection not established');
end



% Storing the images and the image names in oracle table Images

exdata1 = {FileName, img};
fastinsert(conn, 'images', {'imagename';'image'},...
exdata1)


% Storing the image names and gray level feature vector in oracle table
% graylevelfv

exdata2 = {FileName, frequency(1), frequency(2), frequency(3),...
```

```
    frequency(4), frequency(5), frequency(6), frequency(7),...
    frequency(8),frequency(9), frequency(10), frequency(11),...
    frequency(12), frequency(13), frequency(14), frequency(15),...
    frequency(16)};


fastinsert(conn, 'graylevelfv', {'imagename'; 'firstbin'; 'secondbin';...
    'thirdbin'; 'fourthbin'; 'fifthbin'; 'sixthbin'; 'seventhbin';...
    'eighthbin'; 'ninethbin'; 'tenthbin'; 'eleventhbin'; 'twelvethbin';...
    'thirteenthbin'; 'fourteenthbin'; 'fifteenthbin'; ...
    'sixteenthbin'},exdata2)

% Storing the image names and texture feature vectors in oracle table
% texturefv

exdata3 = {FileName, mean_contrast, mean_correlation,...
    mean_homogeneity, mean_energy};
fastinsert(conn, 'texturefv', {'imagename';'contrast';'correlation';...
    'homogeneity';'energy'}, exdata3);

% Storing the image names and shape feature vector in oracle table shapefv

exdata4 = {FileName, perimeter, area, metric};
fastinsert(conn, 'shapefv', {'imagename';'perimeter';'area';'metric'},...
    exdata4);

exec(conn,'commit');
close (conn);
end



% --- Executes on button press in close2.
function close2_Callback(hObject, eventdata, handles)
% hObject    handle to close2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

pos_size = get(handles.figure1,'Position');

user_response = confirm_exit('Title','Confirm close');

switch user_response
    case {'No'}
        % do nothing
    case 'Yes'
        delete(handles.figure1)
end
```

image_search.m (Searching an image in the database)

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

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

% Last Modified by GUIDE v2.5 13-Mar-2011 01:10:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @image_search_OpeningFcn, ...
                   'gui_OutputFcn',  @image_search_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before image_search is made visible.
function image_search_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 image_search (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes image_search wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = image_search_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;


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

% Find the value of the slider
value1 = get(handles.slider1,'value');

% Flow thw value in the text field
str = sprintf('%3.1f', value1);
set(handles.label1,'string',str)

set(handles.slider1,'userdata',value1)


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

% Find the value of the slider
value2 = get(handles.slider2,'value');
%
% Flow thw value in the text field
str = sprintf('%3.1f', value2);
set(handles.label2,'string',str)

set(handles.slider2,'userdata',value2)


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

% Find the value of the slider
value3 = get(handles.slider3,'value');

% Flow thw value in the text field
str = sprintf('%3.1f', value3);
set(handles.label3,'string',str)

set(handles.slider3,'userdata',value3)


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function label1_Callback(hObject, eventdata, handles)
% hObject    handle to label1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of label1 as text
%        str2double(get(hObject,'String')) returns contents of label1 as a
double


% --- Executes during object creation, after setting all properties.
function label1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to label1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

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



function label2_Callback(hObject, eventdata, handles)
% hObject    handle to label2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of label2 as text
%        str2double(get(hObject,'String')) returns contents of label2 as a
double


% --- Executes during object creation, after setting all properties.
function label2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to label2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

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

```
function label3_Callback(hObject, eventdata, handles)
% hObject    handle to label3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of label3 as text
%        str2double(get(hObject,'String')) returns contents of label3 as a
double


% --- Executes during object creation, after setting all properties.
function label3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to label3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

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


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

value1 = get(handles.slider1,'userdata');
value2 = get(handles.slider2,'userdata');
value3 = get(handles.slider3,'userdata');

finalvalue = value1 + value2 + value3;
% assignin('base','va',finalvalue);

if finalvalue >= 1.1314
    errordlg('Wrong wightages Assigned to Features','Assignment Error');
end

if finalvalue < 0.8700
    errordlg('Wrong wightages Assigned to Features','Assignment Error');
end

% --- Executes on button press in search.
function search_Callback(hObject, eventdata, handles)
% hObject    handle to search (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

FileName = get(handles.pushbutton1,'userdata');
fullName = get(handles.pushbutton1,'userdata');
```

```
img = imread(fullName);

qimage = struct('Name',FileName,'Images',img);
qimage.Name;
qimage.Images;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%  CALCULATING GRAY LEVEL FEATURE VECTORS OF THE QUERY IMAGE %%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x_axis = 0:16:255;
y_axis = img(1:end);
% hist(y_axis,x_axis);

frequency = histc(y_axis,x_axis);

qgraylevelFV = struct('Name',FileName,'Freq',frequency);
qgraylevelFV.Name;
qgraylevelFV.Freq(1);
qgraylevelFV.Freq(2);
qgraylevelFV.Freq(3);
qgraylevelFV.Freq(4);
qgraylevelFV.Freq(5);
qgraylevelFV.Freq(6);
qgraylevelFV.Freq(7);
qgraylevelFV.Freq(8);
qgraylevelFV.Freq(9);
qgraylevelFV.Freq(10);
qgraylevelFV.Freq(11);
qgraylevelFV.Freq(12);
qgraylevelFV.Freq(13);
qgraylevelFV.Freq(14);
qgraylevelFV.Freq(15);
qgraylevelFV.Freq(16);

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%  CALCULATING TEXTURE FEATURE VECTORS OF THE QUERY IMAGE  %%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

GLCMS = graycomatrix(img,'Offset',[0 1; 0 2; 0 3; 0 4;-1 1; -2 2; -3 3;...
    -4 4;
-1 0; -2 0; -3 0; -4 0;-1 -1; -2 -2; -3 -3; -4 -4]);

contrast = graycoprops(GLCMS,'contrast');
correlation = graycoprops(GLCMS,'correlation');
energy = graycoprops(GLCMS,'energy');
homogeneity = graycoprops(GLCMS,'homogeneity');

% Calculating mean of all the contrast of 16 GLCMS

sum_contrast = 0;
for i = 1:16
    sum_contrast = sum_contrast + contrast.Contrast(i);
    mean_contrast = sum_contrast/16;
end
mean_contrast;

% Calculating mean of all the correlation of 16 GLCMS
```

```
sum_correlation = 0;
for i = 1:16
    sum_correlation = sum_correlation + correlation.Correlation(i);
    mean_correlation = sum_correlation/16;
end
mean_correlation;

% Calculating mean of all the homogeneity of 16 GLCMS

sum_homogeneity = 0;
for i = 1:16
    sum_homogeneity = sum_homogeneity + homogeneity.Homogeneity(i);
    mean_homogeneity = sum_homogeneity/16;
end
mean_homogeneity;

% Calculating mean of all the energy of 16 GLCMS

sum_energy = 0;
for i = 1:16
    sum_energy = sum_energy + energy.Energy(i);
    mean_energy = sum_energy/16;
end
mean_energy;

qtextureFV =  struct('Name',FileName,'Avgcontrast',mean_contrast,...
    'Avgcorrelation',mean_correlation,'Avghomogeneity',mean_homogeneity,...
    'Avgenergy',mean_energy);

qtextureFV.Name;
qtextureFV.Avgcontrast;
qtextureFV.Avgcorrelation;
qtextureFV.Avghomogeneity;
qtextureFV.Avgenergy;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%CALCULATING SHAPE FEATURE VECTORS OF THE QUERY IMAGE  %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Convert input image into binary image using threshold value

binary_image = im2bw(img,0.94);
% figure, imshow(binary_image);

% Choosing the structuring element

se = strel('disk',3);

% Morphological operation for image smoothing

openedBW = imopen(binary_image,se);
% figure, imshow(openedBW);



bw = bwareaopen(openedBW,1500);
% figure, imshow(bw)
```

```
% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
bw = imfill(bw,'holes');

% figure,imshow(bw)

[B,L] = bwboundaries(bw,'noholes');

stats = regionprops(L,'Area','Centroid');

threshold = 0.94;

% loop over the boundaries
for k = 1:length(B)

  % obtain (X,Y) boundary coordinates corresponding to label 'k'
  boundary = B{k};

  % compute a simple estimate of the object's perimeter
  delta_sq = diff(boundary).^2;
  perimeter = sum(sqrt(sum(delta_sq,2)));

  % obtain the area calculation corresponding to label 'k'
  area = stats(k).Area;

  % compute the roundness metric
  metric = 4*pi*area/perimeter^2;

  qshapeFV = struct('Name',FileName,'Peri',perimeter,'Ar',area,...
      'Metr',metric);

  qshapeFV.Name;
  qshapeFV.Peri;
  qshapeFV.Ar;
  qshapeFV.Metr;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%    ESTABLISHING CONNECTION BETWEEN ORACLE AND MATLAB    %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


img = reshape(img,[],1);
javaaddpath 'D:/oracle/product/10.2.0/db_1/jdbc/lib/ojdbc14.jar';
conn = database('oracle','system','deepak',...
   'oracle.jdbc.driver.OracleDriver', ....
   'jdbc:oracle:thin:');

a = isconnection(conn);
if a == 0
    disp ('connection not established');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%   GRAYLEVEL SIMILARITY CALCULATION     %%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


curs2 = exec(conn, 'select * from graylevelfv');
setdbprefs('DataReturnFormat','structure');

curs2 = fetch(curs2);
temp = curs2.Data;

count = rows(curs2);
temp = curs2.Data;

p = 1;
while p < count + 1
    temp.FIRSTBIN(p);
    temp.SECONDBIN(p);
    temp.THIRDBIN(p);
    temp.FOURTHBIN(p);
    temp.FIFTHBIN(p);
    temp.SIXTHBIN(p);
    temp.SEVENTHBIN(p);
    temp.EIGHTHBIN(p);
    temp.NINETHBIN(p);
    temp.TENTHBIN(p);
    temp.ELEVENTHBIN(p);
    temp.TWELVETHBIN(p);
    temp.THIRTEENTHBIN(p);
    temp.FOURTEENTHBIN(p);
    temp.FIFTEENTHBIN(p);
    temp.SIXTEENTHBIN(p);

    glsimilarity = (sqrt ((temp.FIRSTBIN(p)-qgraylevelFV.Freq(1))^2 +...
        (temp.SECONDBIN(p)-qgraylevelFV.Freq(2))^2 +...
        (temp.THIRDBIN(p)-qgraylevelFV.Freq(3))^2 +...
        (temp.FOURTHBIN(p)-qgraylevelFV.Freq(4))^2 +...
        (temp.FIFTHBIN(p)-qgraylevelFV.Freq(5))^2 +...
        (temp.SIXTHBIN(p)-qgraylevelFV.Freq(6))^2 +...
        (temp.SEVENTHBIN(p)-qgraylevelFV.Freq(7))^2 +...
        (temp.EIGHTHBIN(p)-qgraylevelFV.Freq(8))^2 +...
        (temp.NINETHBIN(p)-qgraylevelFV.Freq(9))^2 +...
        (temp.TENTHBIN(p)-qgraylevelFV.Freq(10))^2 +...
        (temp.ELEVENTHBIN(p)-qgraylevelFV.Freq(11))^2 +...
        (temp.TWELVETHBIN(p)-qgraylevelFV.Freq(12))^2 +...
        (temp.THIRTEENTHBIN(p)-qgraylevelFV.Freq(13))^2 +...
        (temp.FOURTEENTHBIN(p)-qgraylevelFV.Freq(14))^2 +...
        (temp.FIFTEENTHBIN(p)-qgraylevelFV.Freq(15))^2 +...
        (temp.SIXTEENTHBIN(p)-qgraylevelFV.Freq(16))^2));

        % accesing the gray level similarity weightage

        value1 = get(handles.slider1,'userdata');

        assignin('base','test1',value1);
        % calculating the gray level weightage similarity



       weighted_gl_similarity(p) = value1 * glsimilarity;
       assignin('base','testa',weighted_gl_similarity);
```

```
        weighted_gl_similarity(p) = [weighted_gl_similarity(p)];
        image_name(p) = temp.IMAGENAME(p);
        assignin('base','test1',weighted_gl_similarity);

        p = p + 1;


end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%   TEXTURE SIMILARITY CALCULATION     %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

curs3 = exec(conn, 'select * from texturefv');
setdbprefs('DataReturnFormat','structure');

curs3 = fetch(curs3);
temp = curs3.Data;

count3 = rows(curs3);
p = 1;
while p < count3 + 1
    temp.CONTRAST(p);
    temp.CORRELATION(p);
    temp.HOMOGENEITY(p);
    temp.ENERGY(p);

    texsimilarity = sqrt ((temp.CONTRAST(p)-qtextureFV.Avgcontrast)^2 +...
        (temp.CORRELATION(p)-qtextureFV.Avgcorrelation)^2 +...
        (temp.HOMOGENEITY(p)-qtextureFV.Avghomogeneity)^2 +...
        (temp.ENERGY(p)-qtextureFV.Avgenergy)^2);

    % accesing the gray level similarity weightage

    value2 = get(handles.slider2,'userdata');

   assignin('base','test2',value2);

    % calculating the weighted similarity

    weighted_tex_similarity(p) = value2 * texsimilarity;
    assignin('base','testb',weighted_tex_similarity);

    weighted_tex_similarity(p) = [weighted_tex_similarity(p)];
    image_name(p) = temp.IMAGENAME(p);
    assignin('base','test2',weighted_tex_similarity);

    p = p +1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%   SHAPE SIMILARITY CALCULATION     %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

curs4 = exec(conn, 'select * from shapefv');
```

```
setdbprefs('DataReturnFormat','structure');

curs4 = fetch(curs4);
 temp = curs4.Data;

count4 = rows(curs4);
p = 1;
while p < count4 + 1
    temp.PERIMETER(p);
    temp.AREA(p);
    temp.METRIC(p);


    shapesimilarity = sqrt ((temp.PERIMETER(p)-qshapeFV.Peri)^2 +...
        (temp.AREA(p)-qshapeFV.Ar)^2 +...
        (temp.METRIC(p)-qshapeFV.Metr)^2);

     % accesing the gray level similarity weightage

    value3 = get(handles.slider3,'userdata');

   assignin('base','test3',value3);

    % calculating the weighted similarity

    weighted_shape_similarity(p) = value3 * shapesimilarity;

    assignin('base','testc',weighted_shape_similarity);
    weighted_shape_similarity(p) = [weighted_shape_similarity(p)];
    imagename_array(p) = temp.IMAGENAME(p);
    assignin('base','test3',imagename_array);
    assignin('base','test4',weighted_shape_similarity);


    value_array(p)= weighted_gl_similarity(p) +...
        weighted_tex_similarity(p)+...
        weighted_shape_similarity(p);
    assignin('base','test5',value_array);

    p = p +1;

end

%%%% SORTING THE SIMILARITY VALUES AND ASSIGNING TO IMAGE MAMES   %%%%%%%%

value_array;
imagename_array;

aa = 1;
bb = 1;

for aa = 1:length(value_array)
    if aa <= length(value_array)
        for bb = 1:length(imagename_array)
            if bb <= length(imagename_array)
                if value_array(aa) <= value_array(bb)
                    tempo = value_array(aa);
                    value_array(aa) = value_array(bb);
```

```
                    value_array(bb) = tempo;
                    tmp = imagename_array(aa);
                    imagename_array(aa) = imagename_array(bb);
                    imagename_array(bb) = tmp;
                end
            end
        end
    end
end


for bb = 1:length(value_array)
    value_array(bb);
end

for bb = 1:length(value_array)
    imagename_array(bb);
end

value_array;
imagename_array;


folder_images = dir('*.tif');
number_of_images = length(folder_images);
for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(1))
        imagename_array(1)
        image1 = cur_image;
        imshow(image1,'Parent',handles.axes2);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(2))
        imagename_array(2)
        image2 = cur_image;
        imshow(image2,'Parent',handles.axes3);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(3))
        imagename_array(3)
        image3 = cur_image;
        imshow(image3,'Parent',handles.axes4);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(4))
```

```
        imagename_array(4)
        image4 = cur_image;
        imshow(image4,'Parent',handles.axes5);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(5))
        imagename_array(5)
        image5 = cur_image;
        imshow(image5,'Parent',handles.axes6);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(6))
        imagename_array(6)
        image6 = cur_image;
        imshow(image6,'Parent',handles.axes7);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(7))
        imagename_array(7)
        image6 = cur_image;
        imshow(image6,'Parent',handles.axes8);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(8))
        imagename_array(8)
        image6 = cur_image;
        imshow(image6,'Parent',handles.axes9);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(9))
        imagename_array(9)
        image6 = cur_image;
        imshow(image6,'Parent',handles.axes10);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(10))
```

```
        imagename_array(10)
        image6 = cur_image;
        imshow(image6,'Parent',handles.axes11);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(11))
        imagename_array(11)
        image6 = cur_image;
        imshow(image6,'Parent',handles.axes12);
    end
end

for x = 1:number_of_images
    cur_imagename = folder_images(x).name;
    cur_image = imread(cur_imagename);
    if strcmp(cur_imagename,imagename_array(12))
        imagename_array(12)
        image6 = cur_image;
        imshow(image6,'Parent',handles.axes13);
    end
end

close(curs2)
close(curs3)
close(curs4)

exec(conn,'commit');
close (conn);

end

% --- Executes on button press in close3.
function close3_Callback(hObject, eventdata, handles)
% hObject    handle to close3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% to pass to the modal dialog.
pos_size = get(handles.figure1,'Position');
% Call modaldlg with the argument 'Position'.
user_response = confirm_exit('Title','Confirm Close');
switch user_response
    case {'No'}
        % take no action
    case 'Yes'
        % Prepare to close GUI application window
        delete(handles.figure1)
end



% --- Executes on button press in select.
function select_Callback(hObject, eventdata, handles)
% hObject    handle to select (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
[FileName,folder] = uigetfile({'*.tif*'},'Select file');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
J = imread(fullName);
imshow(J,'Parent',handles.axes1);
set(handles.pushbutton1,'userdata',fullName)
set(handles.pushbutton1,'userdata',FileName)
guidata(hObject,handles);

% --- Executes on button press in Back1.
function Back1_Callback(hObject, eventdata, handles)
% hObject    handle to Back1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

homepage;


% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes2


% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes3


% --- Executes during object creation, after setting all properties.
function axes4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes4


% --- Executes during object creation, after setting all properties.
function axes5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes5


% --- Executes during object creation, after setting all properties.
function axes6_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to axes6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes6


% --- Executes during object creation, after setting all properties.
function axes7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes7
```

confirm_exit.m (close the user interface window)

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

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

% Last Modified by GUIDE v2.5 25-Feb-2011 07:34:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @confirm_exit_OpeningFcn, ...
                   'gui_OutputFcn',  @confirm_exit_OutputFcn, ...
```

```
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before confirm_exit is made visible.
function confirm_exit_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 confirm_exit (see VARARGIN)

% Choose default command line output for confirm_exit
handles.output = 'Yes';

% Update handles structure
guidata(hObject, handles);

% Insert custom Title and Text if specified by the user
% Hint: when choosing keywords, be sure they are not easily confused
% with existing figure properties.  See the output of set(figure) for
% a list of figure properties.
if(nargin > 3)
    for index = 1:2:(nargin-3),
        if nargin-3==index, break, end
        switch lower(varargin{index})
         case 'title'
          set(hObject, 'Name', varargin{index+1});
         case 'string'
          set(handles.text1, 'String', varargin{index+1});
        end
    end
end

% Determine the position of the dialog - centered on the callback figure
% if available, else, centered on the screen
FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);
if isempty(gcbf)
    ScreenUnits=get(0,'Units');
    set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');
    set(0,'Units',ScreenUnits);

    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);
    FigPos(2)=2/3*(ScreenSize(4)-FigHeight);
else
```

```
    GCBFOldUnits = get(gcbf,'Units');
    set(gcbf,'Units','pixels');
    GCBFPos = get(gcbf,'Position');
    set(gcbf,'Units',GCBFOldUnits);
    FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
                   (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);
set(hObject, 'Units', OldUnits);

% Show a question icon from dialogicons.mat - variables questIconData
% and questIconMap
load dialogicons.mat

IconData=questIconData;
questIconMap(256,:) = get(handles.figure1, 'Color');
IconCMap=questIconMap;

Img=image(IconData, 'Parent', handles.axes1);
set(handles.figure1, 'Colormap', IconCMap);

set(handles.axes1, ...
    'Visible', 'off', ...
    'YDir'   , 'reverse'       , ...
    'XLim'   , get(Img,'XData'), ...
    'YLim'   , get(Img,'YData')  ...
    );

% Make the GUI modal
set(handles.figure1,'WindowStyle','modal')

% UIWAIT makes confirm_exit wait for user response (see UIRESUME)
uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = confirm_exit_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;

% The figure can be deleted now
delete(handles.figure1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.output = get(hObject,'String');

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

```
% Use UIRESUME instead of delete because the OutputFcn needs
% to get the updated handles structure.
uiresume(handles.figure1);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.output = get(hObject,'String');

% Update handles structure
guidata(hObject, handles);

% Use UIRESUME instead of delete because the OutputFcn needs
% to get the updated handles structure.
uiresume(handles.figure1);


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(handles.figure1, 'waitstatus'), 'waiting')
    % The GUI is still in UIWAIT, us UIRESUME
    uiresume(handles.figure1);
else
    % The GUI is no longer waiting, just close it
    delete(handles.figure1);
end


% --- Executes on key press over figure1 with no controls selected.
function figure1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Check for "enter" or "escape"
if isequal(get(hObject,'CurrentKey'),'escape')
    % User said no by hitting escape
    handles.output = 'No';

    % Update handles structure
    guidata(hObject, handles);

    uiresume(handles.figure1);
end

if isequal(get(hObject,'CurrentKey'),'return')
    uiresume(handles.figure1);
end
```