Cape Peninsula
University of Technology

IMAGE CLASSIFICATION, STORAGE AND RETRIEVAL SYSTEM FOR A 3 U CUBESAT

By

JEAN MARIE GASHAYIJA

Thesis submitted in fulfillment of the requirements for the degree

Master of Technology: Electrical Engineering

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Prof. E Biermann

Bellville

December 2014

DECLARATION

I, Jean Marie Gashayija, hereby declare that the contents of this dissertation/thesis represent my own unaided work, and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed **Date**

ABSTRACT

Small satellites, such as CubeSats are mainly utilized for space and earth imaging missions. Imaging CubeSats are equipped with high resolution cameras for the capturing of digital images, as well as mass storage devices for storing the images. The captured images are transmitted to the ground station and subsequently stored in a database.

The main problem with stored images in a large image database, identified by researchers and developers within the last number of years, is the retrieval of precise, clear images and overcoming the semantic gap. The semantic gap relates to the lack of correlation between the semantic categories the user requires and the low level features that a content-based image retrieval system offers. Clear images are needed to be usable for applications such as mapping, disaster monitoring and town planning.

The main objective of this thesis is the design and development of an image classification, storage and retrieval system for a CubeSat. This system enables efficient classification, storing and retrieval of images that are received on a daily basis from an in-orbit CubeSat. In order to propose such a system, a specific research methodology was chosen and adopted. This entails extensive literature reviews on image classification techniques and image feature extraction techniques, to extract content embedded within an image, and include studies on image database systems, data mining techniques and image retrieval techniques. The literature study led to a requirement analysis followed by the analyses of software development models in order to design the system.

The proposed design entails classifying images using content embedded in the image and also extracting image metadata such as date and time. Specific features extraction techniques are needed to extract required content and metadata. In order to achieve extraction of information embedded in the image, colour feature (colour histogram), shape feature (Mathematical Morphology) and texture feature (GLCM) techniques were used.

Other major contributions of this project include a graphical user interface which enables users to search for similar images against those stored in the database. An automatic image extractor algorithm was also designed to classify images according to date and time, and colour, texture and shape features extractor techniques were proposed. These ensured that when a user wishes to query the database, the shape objects, colour quantities and contrast contained in an image are extracted and compared to those stored in the database.

Implementation and test results concluded that the designed system is able to categorize images automatically and at the same time provide efficient and accurate results. The features extracted for each image depend on colour, shape and texture methods. Optimal values were also incorporated in order to reduce retrieval times. The mathematical morphological technique was used to compute shape objects using erosion and dilation operators, and the co-occurrence matrix was used to compute the texture feature of the image.

Keywords: Image classification, CubeSat, Content based image retrieval system, Nanosatellite, feature extraction and distance measure.

ACKNOWLEDGEMENTS

I wish to thank:

- My wife, Ruth Francis Gashayija, for the endurance, support, understanding and encouragement during the time of study. Without her advice and moral support, things would have not been the same.
- My supervisor, Professor Elmarie Biermann, for her supervision, guidance, advice and contribution throughout the thesis write up. To her, I say Thank you very much.
- To the Gashayija family for moral support and financial support throughout this thesis.
- The Roger Family for editing my work, advice, financial support and educating me for all my tertiary education. I say no word can express how much I feel. Thank you so much for this great opportunity given me.
- My colleague, Paul Bayendang, and others for their support and advice.
- The French South African Institute of Technology for their financial assistance towards this research. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to the French South African Institute of Technology.

DEDICATION

I wish to dedicate this thesis to late Engineer Michael, my sisters Esther, Janet, Christine and all brothers Cassien, Safari, Emmanuel and Seth for motivating when times were tough.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF LISTINGS

LIST OF ABBREVIATIONS

1 U	One Unit CubeSat
3 U	Three Unit CubeSat
ADCS	Attitude Determination Control System
ANN	Artificial Neural Network
BMP	Bitmap File Format
CBIR	Content Based Image Retrieval
CPUT	Cape Peninsula University of Technology
CCIP	Paris Chamber of Commerce and Industry
DT	Decision Tree
EM	Expectation Maximization
F'SATI	French South African Institute of Technology
GCH	Global Colour Histograms
GIF	Graphics Interchange Format
GLCM	Gray level Co-occurrence Matrix
HMO	Hermanus Magnetic Observatory
HSB	Hue Saturation and Brightness
HSV	Hue Saturation and Value
JPEG	Joint Photographic Experts Group
KNN	K-Nearest Neighbour
LCH	Local Colour Histograms
NB	Naive Bayesian
NRF	National Research Foundation
OAO	One against One
ODBC	Open Database Connectivity
PPOD	Poly Picosat Orbital Deployer
RGB	Red Green and Blue
SANSA	South Africa National Space Agency
SOM	Self Organizing Map
SPIE	International Society for optical Photonics and Imaging engine
SVM	Support Vector Machine
TUT	Tshwane University of Technology
UPEC	University of Paris Est Creteil

CHAPTER 1

INTRODUCTION

1.1 Introduction

"CubeSats are now considered a disruptive technology revolutionizing the way satellites are built. Most large satellites involve development of custom equipment while CubeSats mostly rely on Commercial of the shelf components (COTS) due to modest budgets available for their development." (Aslan, Sofyali, Umit, Tola, Oz & Gulgonul, 2011).

Low cost and reduced development time of small satellites (and especially CubeSats) have encouraged academic institutions to participate in space activities. These development platforms have motivated the establishment of small satellite programs at Universities and emerging space industries throughout the world. Imaging of Earth is a common mission for most small satellites, as such images are a means of obtaining and conveying information regarding the planet. Pictures convey concise information such as position, size, and inter-relationships of objects, as well as portray spatial information that can be recognized as objects (Sivakumar, Roy, Harmsen & Saha, 2003).

An imaging CubeSat is equipped with an on-board camera payload for capturing digital images as well as a mass storage device for storage purposes. The imager forms part of the imaging payload subsystem onboard the CubeSat and can take pictures of any point on Earth if the orbit permits. On-board mass memory is required to store the large amounts of acquired image data. In commercial satellites, hard disk drives are used, while in small satellites flash memory on Secure Digital (SD) cards are utilized. The captured images are transmitted to the ground station and usually stored in a database for further processing. Advanced processing of the images need to be performed in order to identify and extract precise and clear images to be usable for applications such as mapping, disaster monitoring and weather (Khurshid, Mahmood & Islam, 2013; Sivakumar *et al.,* 2003).

An image database system is a combination of storage and processing systems that facilitates the handling of images by structuring them and allowing for easier querying. One of the challenges for an imaging CubeSat mission is the formulation of an image database system that enables precise retrieval of specific images for the user at a low cost. The main focus of this thesis lies within the Content based image retrieval (CBIR) system design to ease the retrieval, storage and querying of images.

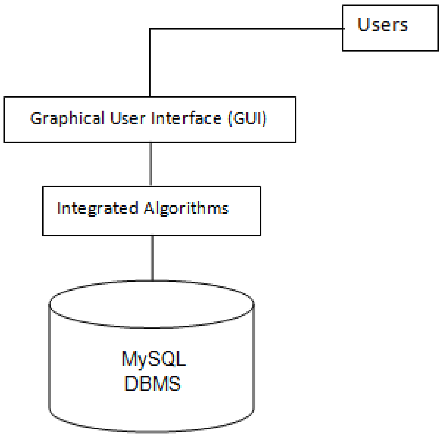


Figure 1.1: CBIR User's system architecture

In Figure 1.1, the overall CBIR User system architecture consists of users, a graphical user interface, integrated algorithms and a database management system. This architecture is based on CBIR techniques that allow users to search, store and retrieve images. Typically, a CBIR system has two phase processes: processing of images and storing images into database which can be queried. In CBIR, queries are usually issued by giving an image example or by starting with a random image from a current image database. In most CBIR systems, the first stage deals with the actual populating database. The relevant image content is extracted and indices are built in the database. An interface is given to the end users to allow them to query the image database in a variety of ways. In CBIR end users are allowed to search or query the database using image contents such as colour, texture and shape contents, and this is called query by example. This is whereby a reference image is used as input so that similar image is returned. To compare the database of stored indices, a query index has to be constructed i.e. a reference image. A distance measure is used to compute the similarity between the query against stored database images. The calculated value or measure of similarity is used to rank the validity of a given image. The similarity ranking is used to select the best group of images that the user is interested in, and the retrieval is based on the similarities calculated from their indices. In simple terms, when end users query the database to find similar images using image content, the algorithms take the given reference image and compares it to those stored images in the database. The other role of the algorithms is to ensure both query and stored images are extracted before comparing them.

1.2 Background to the project

The French South Africa Institute of Technology (F'SATI) is a graduate school focusing on the training of graduate students in the broad fields of electrical engineering and information and communication technologies. F'SATI was established in 1996, through collaboration between the French and South African Governments. A number of players, such as the National Research Foundation (NRF), the Tshwane University of Technology (TUT), the Paris Chamber of Commerce and Industry (CCIP), and the Cape Peninsula University of Technology (CPUT), are committed to the establishment of an advanced research unit (F'SATI, 2012). F'SATI has two branches in South Africa, namely the Pretoria branch located at TUT and the Cape Town branch on the premises of CPUT. They offer graduate programmes on Master and Doctorate levels in partnership with certain universities in France such as L' Ecole De L' Innovation Technologique (ESIEE) Paris and the University of Paris Est Creteil (UPEC). F'SATI, in collaboration with the South African National Space Agency (SANSA) and the Hermanus Magnetic Observatory (HMO), are designing a series of CubeSats, 1U and 3U respectively. The 1U CubeSat was successfully launched in 2014 and the 3U CubeSat is set for completion in 2015.

1.3 The CubeSat Standard

As described by Horais and Twiggs (2001), the CubeSat was originally proposed by Proffessors Jordi Puig-Suari of California Polytechnic State University and Bob Twiggs of Stanford University. They introduced a general standard CubeSat specification: a $10cm^3$ shape with mass of no more than 1kg (as illustrated in Figure 1.2). CubeSats are being utilized for various missions such as military, scientific as well communication purposes.

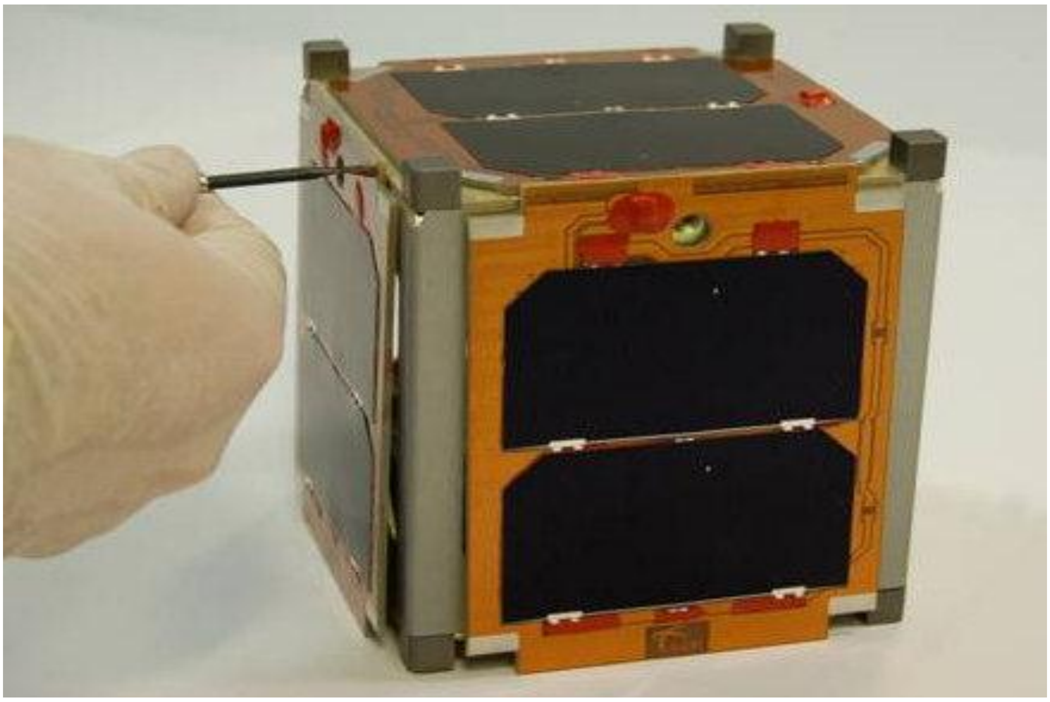


Figure 1.2: CubeSat Model

(Adapted from Clyde Space, 2012)

The standard 1U CubeSat consists of five subsystems, namely power, communication, on-board computer (OBC), attitude determination and control system, and camera payload subsystem. The attitude determination and control system maintains CubeSat stabilization and orientation of the desired direction and position to enable the camera to take images at certain positions and locations (Schor, Scowcroft, Nichols & Kinsner, 2009).

According to Schor *et al.* (2009), satellites are normally classified according to their mass. For example, small satellites weigh less than 500kg, medium satellites weigh between 500kg-1000kg and large satellites weigh more than 1000kg. Small satellites are further divided into five subcategories according to their weight. Mini satellites weigh between 100kg-500kg, micro satellites between 10kg-100kg, nano-satellites between 1kg-10kg, Pico satellites between 0.1kg-1kg and Femto satellite less than 0.1kg. CubeSats fall within the nano-satellites category and may consist of a number of units, such as 1U or 3U. One unit constitutes a standard $10cm^3$, and larger versions are formed by stacking units together. The maximum number of CubeSats that can be stacked and still fit into the Poly Picosat Orbital Deployer (P-POD) is the 3U CubeSat (F'SATI, 2012).

A CubeSat ground station is an important entity within the satellite environment. The purpose of a ground station is to conduct radio frequency propagation experiments, track and communicate with various orbiting satellites, as well as to commission and operate the CubeSats in orbit. The 1U and 3U CubeSats developed at F'SATI are controlled via a ground station located on the roof of the Electrical Engineering building in Bellville Campus at Cape Peninsula University of Technology (CPUT) in Cape Town. The CubeSats are launched into polar orbit at an altitude of between 450km and 650km. The 1U CubeSat (ZACUBE01) contains a camera (0.3 megapixels), payload and a beacon transmitter. Its primary mission is to use a beacon transmitter to characterize ionosphere radar antennas in Antarctica and the on-board camera is used to capture images of Earth. The 3U CubeSat (ZACUBE02) includes a five megapixels (high resolution) camera and will focus on capturing images for specific applications such as fire warning and control.

1.4 CubeSat Transmission Times

The CubeSat orbits the earth once every 90 minutes in low earth orbit of between 600-800km.

The satellite is only visible for about ten minutes to transmit data to a specific ground station, other times it cannot transmit data being received on a ground station due to the satellite being below the horizon. When having a time window of 600 seconds and data rate of 9600 bits per second (bps), the amount of 9600bps x 600s = 5, 760 000 bits or 730 KiloBytes can be transmitted once passing by a ground station. A single colour picture with a spatial resolution of 3 Mega-Pixels and a spectral resolution of 3 x 8 bits gives the total of 3 x 1024 x 1024 x 3 x 8 = 9.44MegaBytes. To be able to transmit a full picture in this ten minute period, the maximum picture size was calculated to not exceed 700KB. In the above example, less that 10% of the picture could be transmitted. In order for the whole picture to be transmitted, the raw data needs to be encoded and compressed. Two issues – of poor quality and better quality – arises when compressing an image. In this instance, a JPEG compression algorithm with a variable compession rate of upto 20:1 is used. With a high compression rate, alot of information is taken away and quality is decreased. The JPEG compression algorithm is a data reduction, real information is reduced for the sake of decreasing the size in order to allow it to be transmitted. If a high resolution picture is desired, i.e. 3Mega-Pixel or 5Mega-Pixel picture, a high compression is required however, the resolution of the picture is reduced. Because of the high compression ratio, alot of information is removed from the picture leading it to be of poor quality (Polaschegg, 2005).

Table 1.1: Compression Rate of transmitting picture to a ground station

Resolution (mega-pixel)	Raw data (kiloByte)	Compression rate needed at	
		9600 pbs	19200 pbs
1	3072	1 : 4.39	1 : 2.19
1.4	4300	1 : 6.14	1 : 3.07
2	6144	1 : 8.78	1 : 4.39
2.4	7372	1 : 10.53	1 : 5.27
3	9216	1 : 13.17	1 : 6.58
5	15360	1 : 21.94	1 : 10.97

Table 1.1 illustrates the compression rates of various image sizes needed to transmit data to a ground station within a period of ten minutes. A picture can be transmitted using either 9600 bits per second or 19200 bits per second. It is useful to transmit a picture to a ground station having enough information for better analyses. However, when a picture is compressed using a compression rate greater than 1:10, alot of information is removed from the original picture and the transmitted picture is not useful. It was observed that when operating at 9600bps, a resolution high than 2 Mega-pixels was not suitable for this cubesat mission. When operating at 19200 bps, a 4 Mega pixel camera could be used, however, operating at 19200bps requires higher power consumption for the camera to take such a high resolution picture. Therefore, the higher the resolution, the higher is the power consumption of the camera. The camera can be switched on and off only when a picture need to be taken, which reduces power consumption. It was noted also only one picture could be taken and transmitted when passing by the ground station (Polaschegg, 2005).

ZACUBE-02 DOWNLINK CALCULATION FOR IMAGES

To transmit data to a ground station using 9600 baud rate and for a 5 Megabyte image, the quickest downlink reception duration is approximately 4000 seconds.

Given parameters:

Downlink data rate or speed: 9.6Kbps max

Image size: 5 Megabytes

Converting 5 Megabytes to bits: $5 \times 10^6 \times 8 = 40 \times 10^6 =$ 40 000 000 bits.

Assuming ideal conditions:

If a maximum of 9600 bits can be transmitted in 1s. Therefore, 40 000 000 bits will take:

40 000 000/9600 = 4166.77 seconds

This is approximately 69.4444 minutes or (~70 minutes).

Therefore, to receive, classify and retrieve the transmitted image after a minimum delay of (~70 minutes) will not be in real time. According to Nzeugaing (2013), in order to achieve real time, CubeSat imagery needed to be compressed to a size of <1.2 KB (which is an acceptable quality). At this compression size, the real time reception can be achieved on downlink data rate of 9600 baud rate. According to Kalman and Reif (2008), 3U CubeSats using thirteen ground stations MISC could provide 196 images per day for a net imaging area of approximately 137 500 km^2.

1.5 Problem Statement

Indexing and categorizing images into semantically meaningful ones using low level feature is a challenging and important problem in content based image retrieval (CBIR). To organize large image collections into categories and provide effective indexing is vital for the browsing and retrieval system. This rather difficult problem has not been adequately addressed in current image database systems. They need to develop a system for indexing and categorizing the huge amounts of data. The other challenge is of providing high level semantic indices for large databases, as users in a CBIR system are typically based on semantics and not on low level features. A successful indexing and categorization of images greatly enhances the performance of content based retrieval systems by filtering out irrelevant classes (Maheswary & Srivastava, 2009; Vailaya, Figueiredo, Jain & Zhang, 2001).

The major problems with stored images in a large image database (an image database is a combination of storage and processing systems that facilitates the handling of images by structuring them and allowing easier querying of the database) are retrieval of precisely, clear images and semantic gap. In context of the CubeSat environment, a large number of images are transferred from the orbiting CubeSat to the ground station on a daily basis. As this number increases with the passing of time as well as with additional CubeSats in orbit, image indexing and categorization becomes vital to ensure applications retrieval are valid and requested images are accurate. The other challenge for such a CubeSat mission is the formulation of a computirized image database which will enable fast and precise retrieval of specific images for the user at low cost.

The motivation for this research is saving on time and cost: in previous years the file system was used to store and retrieve images, which was very costly and time consuming. There was a need for a faster and cheaper database management system, and new technology allows end-users to be able to search, store and retrieve images quickly and at low cost.

1.6 Objectives

The main objective of this thesis is to design and develop an image classification, storage and retrieval system for a CubeSat. This system will enable efficient classification, storing and retrieval of images received from an in-orbit CubeSat.

The specific objectives to be addressed are:

- To study and analyze space and Earth images and study methods to identify specific features.
- To select and propose appropriate image processing techniques to process and extract features in an image.
- To design and propose an image classification scheme suited for images of Earth as captured by orbiting CubeSats.
- To classify images using image classification techniques (i.e. texture classification) once received at the ground station.
- To design a storage and retrieval system.
- To study how images are to be ranked and retrieved from a large database.
- To study techniques to evaluate the performance of an image retrieval system.
- To test, implement and integrate the proposed system for the CubeSat environment.

1.7 Methodology

In order to propose and implement the image classification, storage and retrieval system the following methodology will be followed:

- Literature review on CubeSat imaging missions, whereby information on the CubeSat environment will be collected from related books, journals and academic studies. This will assist in understanding integration of subsystems and especially, the imaging payload.
- Literature review on image database systems and data mining techniques.
- A detailed study on image retrieval systems and what problems they encounter will be investigated.
- Literature review on feature extraction techniques (image processing toolboxes to extract features in an image will be considered).
- Study Matlab as an object oriented programming language, to assist in designing a suitable and friendly graphical user interface for users to search or query images will.

- Study will be conducted on Integrated Development Environment (IDE) to assist with the design and testing of the colour, shape and texture feature extraction algorithms.
- Test and evaluate the accuracy of the proposed design and implementation of the system.
- They are many software development process models, however this study will focus on waterfall, incremental, prototype and spiral models, and choice will be selected based on requirements given in Section 5.3.

1.8 Research significance

CubeSat missions have limited budgets influencing both the design of the space segment and ground segment. The 3U CubeSat (ZACUBE-02) will be used for an imaging mission, hence the need for a low cost efficient storage and retrieval system for the ground station segment of the mission. This research will allow images from ZACUBE-02 to be accessible to users in applications such as land management and location of natural resources. A system that meets the challenging requirements will be delivered and at the same time will provide a simple and less complex image storage and retrieval system for CubeSat missions. Such a Content Based Image Retrieval (CBIR) system will not only benefit space industries, but also general and specialized applications and services such as Google, Flicker and social networks (i.e. Facebook). For example, the proposed system may be useful in hospitals to record and store patients' information. Although, a modification will be needed for entering patients' details such as names, surnames and addresses, and the Graphical User Interface will need to be expanded to accommodate entering details.

1.9 Thesis outline

This thesis is organized as follows: Chapter 2 presents the background concepts, such as CubeSat standards, image compression standards and image feature extraction techniques. Chapter 3 focuses on categories of image classification techniques and provides more detail on supervised classification techniques. Chapter 4 also provides a thorough study of unsupervised classification techniques. Chapter 5 details the requirement analysis and design of the proposed system. Chapter 6 provides the implementation, testing, evaluation and results of the developed system as proposed in Chapter 5. The study is concluded in Chapter 7. Appendix A discusses and lists all codes, while Appendix B provides the test images. Appendix C details the procedures to interface MATLAB and a MySQL database. Appendix D highlights procedures to download and install the database and Appendix E provides descriptions of the MATLAB library components.

1.10 Delimitations and assumptions

The research project focuses on the design of an image classification, storage and retrieval system for a 3U CubeSat. Although images are compressed to alleviate size constraints, this study does not deal with compression techniques and tools. However, a brief elaboration on compression techniques will be included in this work. The 3U Cubesat will have an on-board image compression system, and before transmitting data to the ground station, images will be in a compressed format. Typically, satellite images are captured in 2D or 3D. The conversion between 2D and 3D images falls outside the scope of this thesis.

CHAPTER 2
CUBESAT AND IMAGERY

2.1 Introduction

This chapter presents information related to CubeSats and imagery. This chapter not only aims to introduce CubeSats but also to elaborate on image compression, due to it being part of imagery and image feature extraction techniques.

2.2 The CubeSat

As mentioned in Chapter 1, F'SATI is developing a series of CubeSats; one unit (ZACUBE-01) and three unit (ZACUBE-02). As shown in Figure 2.1, ZACUBE-01 is a student and staff developed satellite, of which development begun in early 2011. Its main payload is a high frequency beacon transmitter (14.099MHz that transmits up to 200mW of RF power) and is used to help characterize the Earth's ionosphere and to calibrate South Africa National Space Agency's (SANSA) Auroral radar installation at the Sanae base in Antarctica. The main objective of the HF beacon on ZACUBE-01 is to provide a continuous radio signal to determine the elevation resolving algorithm of the Super Dual Auroral Radar Network (superDARN). The signal is also used to characterize the beam pattern of this and other HF radar antennas in the SuperDARN network and to characterize the ionosphere over the Polar Regions. It's also equipped with a 0.32MP (565×565) camera (CPUT, 2013).

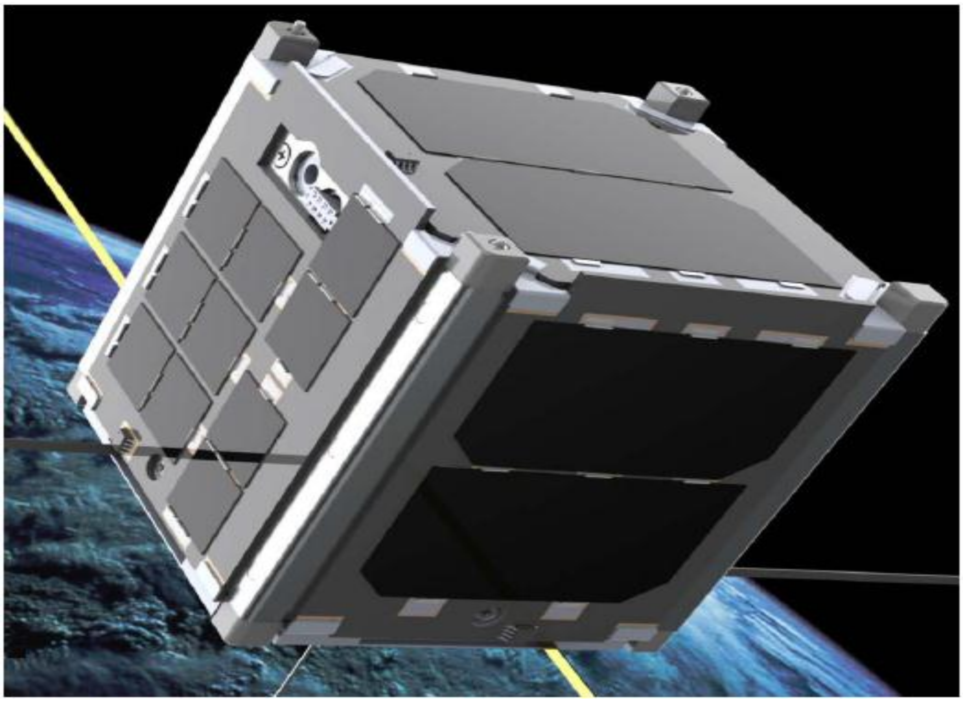


Figure 2.1: The 1U CubeSat
(Adapted from F'SATI, 2012)

ZACUBE-02 is a three unit CubeSat (see Figure 2.2) to be equipped with a high resolution camera of 5MP (2236×2236). A CubeSat consists of a number of subsystems namely power, on-board computer (OBC), payload, communication, as well as the attitude determination and control subsystem. These are described in more detail in the following subsections.

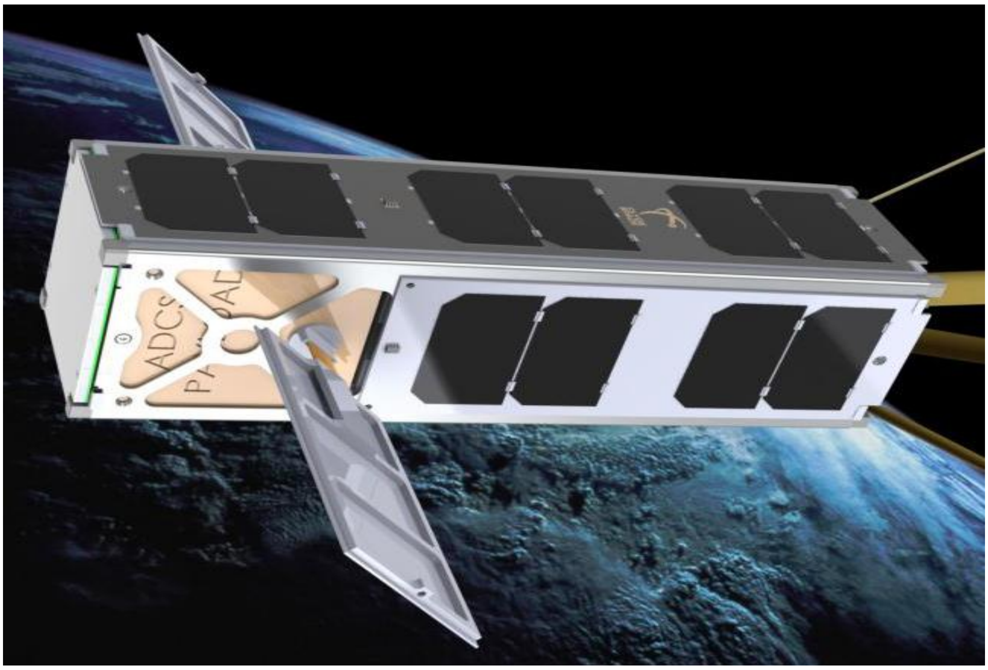


Figure 2.2 : The 3U CubeSat
(Adapted from F'SATI, 2012)

2.2.1 On-board Computer (OBC)

As described by Krishnamurthy (2008), the OBC is the central computing unit of the CubeSat. It interconnects all subsystems, controls and handles, and transfers data between various subsystems. It ensures the satellite comes into direct contact with the ground station for communication purposes. It also monitors the health and status of the satellite, and the operator on the ground station can issue new commands to the satellite through the OBC. All commands and information passes through the OBC before being sent to the relevant subsystem/s. A block diagram detailing the OBC subsystem is shown in Figure 2.3.

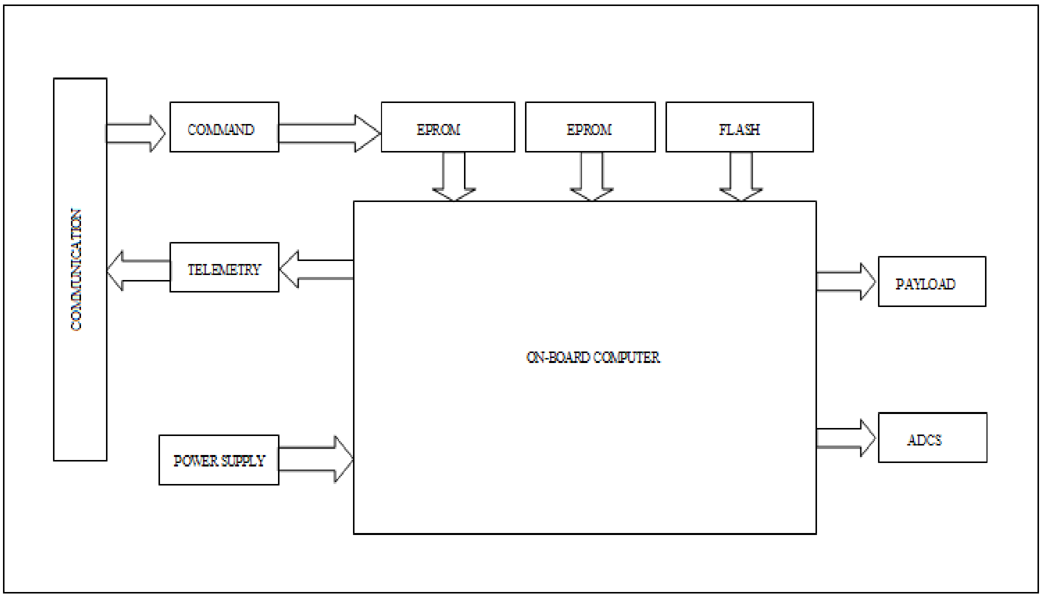


Figure 2.3 : OBC Computer Architecture
(Adapted from AAU CubeSat, 2012)

2.2.2 Electrical Power Subsystem (EPS)

The Electrical power subsystem generates, stores, controls and distributes power to all of the subsystems. The EPS gets its power from solar energy (such as silicon, single junction, dual junction and triple junction), which it stores onto battery. The higher the efficiency of the solar cells, the more power can be delivered from the cells. EPS regulates voltages of all subsystems in the satellite. It offers protection of over-current and over-voltage to all subsystems. The solar cells are configured either in series or parallel using two or three solar cells (Burt, 2011).

2.2.3 Attitude Determination and Control Subsystem (ADCS)

As stated by Larson and Wertz (1999), the ADCS is responsible for stabilization and orientation of the satellite during the mission. The ADCS also ensures that the camera is able to capture images of Earth despite the external disturbance torques (such as gravity gradient torque, solar radiation torque, magnetic disturbance and aerodynamic torque) acting on the satellite.

The ADCS is mainly responsible for the following:

- The stability of the satellite after launch separation (Mohammed, Benyettau, Sweeting & Cooksley, 2005).

- The solar arrays are pointing towards the sun to generate power for the satellite (Sandau, 2003).

- To point payload/s such as cameras and antennas to the desired direction (Scholz, 2003).

- To perform CubeSat attitude maneuver for orbit and payloads operations (Bezold, 2013).
- It is required that the satellite determines its attitude using sensors (such as gyro, sun sensor, star sensor, horizon sensor and magnetometer) and controls it using actuators of the ADCS. The typical actuator that may be used includes reaction wheel, control moment gyro and magnetic torque (Kaplan, 2006).

2.2.4 Communication Subsystem

As stated by Traussnig (2007), the communication subsystem is responsible for sending and receiving information between the satellite and the ground station. Therefore, the success of the satellite mission depends on the ability of the satellite to communicate and exchange data with the ground station.

As stated by Babuscia, Corbin and Clem (2013) most of current CubeSats use antennae, such as dipole, monopole or patch antennae, to transmit or receive data from the satellite to the ground station. Current CubeSat communication systems mostly use Ultra High Frequency (UHF) and S-Band which is equipped with dipole, monopole and patch antennae (achieves maximum of 8dB). CubeSats power consumption usually used to transmit data is approximately 1W.

The communication subsystem of ZACUBE-02 will operate in the 70cm band and will be commanded from the amateur ground station located on top of the Electrical Engineering building at the Bellville Campus of Cape Peninsula University of Technology as shown in Figure 2.4. This ground station uses frequencies of beacon 11.099MHz and VHF/UHF transceiver for uplink 145.86MHz and downlink of 437.345MHz. It will use Omni-directional for the UHF transmitter and the receiver (CPUT, 2013).

Figure 2.4: CPUT Bellville campus ground station
(Adapted from F'SATI, 2012)

2.2.5 Camera Payload Subsystem

ZACUBE-02 contains a 5MP camera for capturing and relaying images. Once the images are captured, it will be stored in volatile memory while awaiting transmission to the ground station. A number of CubeSat missions include image payloads, such as Masat-1 build by Technical University of Budapest in Hungary, where the main payload is a visible camera with a 640×480pixel image area and ground resolution of 2km to 10km. The AAU CubeSat series which was built by Aalborg University in Denmark, uses a CMOS sensor in order to capture the earth's surface in high resolution (Polaschegg, 2005).

Recent CubeSat image payloads missions include ITupSat CubeSat built by Istanbul Technical University in Turkey, with the mission goal to capture colour imagery using CMOS payload (VGA camera based on an OV7620 image sensor) which can be used for better quality video and still image applications. Similar cameras were implemented in CubeSat XI-IV built by Tokyo University in Japan, as well as ICUBE-1 built by the Institute of Space Technology in Pakistan **(**Khurshid, Mahmood & Islam, 2013**)**. Jugnu CubeSat built by the Indian Institute of Technology Kanpur in India is a 3U CubeSat that includes an IR camera and GPS receiver (IITK, 2013). The payload of the CanX-1, built by the University of Toronto Institute for Aerospace Studies in Canada, includes a CMOS sensor with high resolution and consumes less power. It also has other payloads, such asstar-trackers, active three axis magnetic stabilization and GPS based position determination (UTIAS, 2013). The Compass-1, built by Aachen University of Applied science in Germany, focuses on remote sensing and the payload used is a CMOS sensor (OV7648FB Imaging sensor) to take pictures of Earth (Scholz, 2003; Khurshid *et al.,* 2013). The Goliat CubeSat built by University of Bucharest Romania in Romania contains a 3MP optical imager with a 57mm focal length and includes a radiation detector and a micro-meteoroid detector (Klofas & Leveque, 2013). The mission of M-Cubed, built by Michigan State University, is to capture and downlink an image of Earth using a 1.3MP camera (eoportaldirectory, 2013). Almost all designers and developers of CubeSats in the past have incorporated a CMOS camera for image capturing in their designs. Generally in space there are two problems that affect cameras used on satellites; harsh radiation and thermal environments that affect the performance of cameras and endanger the fulfillment of the mission goal of satellite. The image sensors are mainly two types: Charge-coupled devices (CCDs) and Complementary Metal-oxide semiconductor (CMOS) image sensors. The CCDs degrade strongly when exposed to radiation. Yet, CCDs are built with very high performance characteristics. The CMOS image sensors are more robust in irradiated environments. However, CMOS image sensors have slightly worse imaging performance. The CCDs have been used extensively though; they had to add dedicated protective shielding against radiation. Nowadays, CMOS images have improved their performance characteristics and become an option for scientific imaging (De Segura, 2013). Table 2.1 below shows advantages and disadvantages of CCD and CMOS camera.

Table 2.1: Advantages and disadvantages of CCD and CMOS Camera

Camera type	Advantages	Disadvantages
CCD camera	• Higher sensitivity • Electronic shutter without artifacts, • Lower noise • Lower dark current • Smaller pixel size • 100% fill Factor	• Basic function consume several watts (1-5W) • Cannot be monolithically integrated with analog readout and digital control electronics • Cannot guarantee their functionality over the whole temperature range required, • It needs short integration time
CMOS camera	• Low power consumption • Single power supply • High integration capability • Lower cost • Single master clock and • Random access	• Less quantum efficiency, • Image sensors suffer from different noise sources, • Less image quality than CCD.

2.3 Image Compression

The ZACUBE-02 mission is to capture high resolution images of Earth at an altitude of approximately 700km. Due to the large image file size (1MB) and transmission rate, image compression is essential. Image compression decreases image size to allow satellites to partially store and transmit data to the ground station. Typically, a communication data rate between a CubeSat and a ground station is about 9.6 kbps. Two main types of image compression techniques are utilized, namely Lossless compression and Lossy compression. With Lossless compression, this technique is applied on an original image and the compressed image usually looks same as the original image, without loss of information in the converted image or compressed image. Lossy compression reconstructs the image with a varying degree of information loss (Yu, Vladimirova & Sweeting, 2009: 989).

It is required that images captured by ZACUBE-02 should maintain the same image quality before and after compression. In order to maintain image quality, the lossless compression technique will be used on-board the satellite .The mission requirements for ZACUBE-02; it is important to consider image quality, limited storage memory on-board the ZACUBE-02, bandwidth and power consumption of the image processing device. Image compression on-board ZACUBE-02 will be implemented using JPEG 2000 lossless compression. The JPEG 2000 compression is the latest image compression standard to emerge from the Joint Photographic Expert Group. It offers superior compression performance of two decibels compared to other image compressions such as Portable Network Graphics (PNG) and Joint Bi-level Image Expert Group (JBIG) (Syahrul, 2011).

Sparenberg, Bruns and Foessel (2013) proposed using JPEG 2000 compression to compress texture and shape information in an image. Ismailoglu, Benderli, Yesil, Sever, Okcan and Oktem (2005) implemented JPEG 2000 compression onboard the RASAT second small satellite built by Tubitak Space Technologies Research Institute (TUBITAK UZAY) in Turkey to provide high resolution imagery. Both BILSAT-1 and RASAT implemented the JPEG 2000 real time on-board image processing subsystem. Multispectral imaging with spectral bands was compressed and processed simultaneously and the user was able to issue commands from the ground station for changing the compression ratio for each mission. As stated by Ismailoglu, Benderli, Korkmaz, Durna, Kolcak and Tekmen (2002) JPEG 2000 compression uses less memory space and also improves image quality. According to Achary and Tsai (2005), JPEG 2000 compression is an attractive method which is able to transmit high amounts of data to/from ground station given availability of bandwidth in the data links, and this reduces communication costs.

Another lossless compression standard is the JPEG lossless compression standard. It is based on the predictive coding technique. In JPEG lossless compression, two coding techniques are incorporated, namely Huffman and arithmetic coding. When comparing the two JPEG compression modes, lossless and lossy, lossless compression is fully independent of the transform based coding, since it applies differential coding to form the residuals which are then coded via Huffman or arithmetic coding methods. Advantages of JPEG lossless compression are that is easy to implement and simple to compute. Though, when comparing JPEG lossless to JPEG 2000 lossless compression, JPEG 2000 lossless compression provides a better compression technique that is modifiable in different ways to better suit any application. As a measure of image quality, peak signal to noise ratio is used. The JPEG 2000 standard has better peak signal to noise ratio (PSNR) than the JPEG standard (Nzeugaing, 2013).

2.4 Real-time applications

Real-time in context of software engineering perspective, predicts the performance of system rather than just fast processing (Laplante & Neill, 2003). A real-time system should be able to do processing, predict, control and give accurate results. The accuracy of processed results should also consider how much time it takes to complete the task. As stated by Sha, Ragunathan and Sathaye (1994) a real-time system's measurements includes:

- Predictably fast response: the system to predict and provide response in quick time to priority events.
- High degree of schedulability: when the system is dealing with a lot of resources, priority of task to be managed should be properly dealt with and within reasonable time.
- Stability under transient overload: when the system is dealing with high priority tasks, it should guarantee and manage accurately all tasks regardless of whether the system is overloaded.

In processing a lot of images and extracting features in images, real-time image systems are used. To process such huge amount of images, computation and memory resources are need to process the images (Bovik, 2005). In a real-time image system, algorithms can be developed to enhance and extract information contained in each image dimension. The main challenge faced with real-time image systems, is processing and extracting huge amount of data, and this requires computation and memory resources. Figure 2.5 highlights the stages within a real-time image processing system.

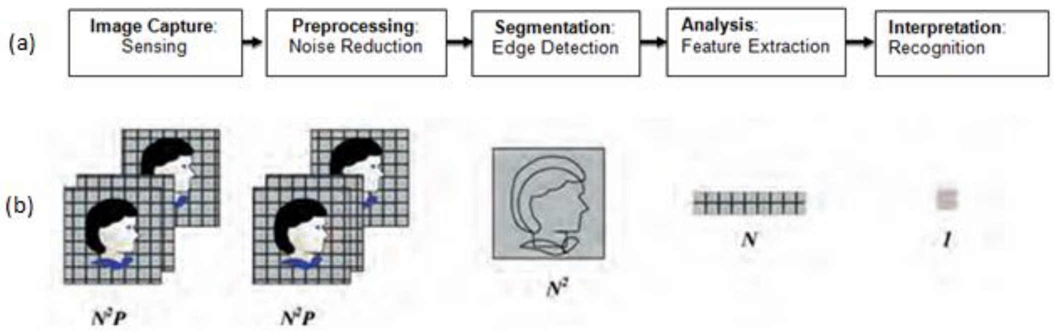


Figure 2.5 : Typical image processing stages:

(a) Typical processing chain (b) Decrease in amount of data processing chain
(Adapted from Kehtarnavaz & Gamadia, 2005)

2.5 Image feature extraction

As stated by Afifi (2011), an image has embedded information, which needs to be extracted. There is a variety of image features (i.e., colour, texture and shape features) which can be utilized to describe an image.

2.5.1 Colour feature

The colour feature technique allows the individual's eyes to recognize or identify objects that are embedded in an image and it also allows the system such image retrieval to extract objects from the same image. In order to extract colour features from an image, a colour system needs to be selected and its properties used. Commonly used colour systems are Red, Green and Blue (RGB), HIS and Hue, and Saturation and Value (HSV). HSV is commonly utilized when converting RGB images to other colour systems (Shih, Huang, Wang, Hung & Kao, 2001). A typical RGB image consists of three components as shown in Figure 2.6.

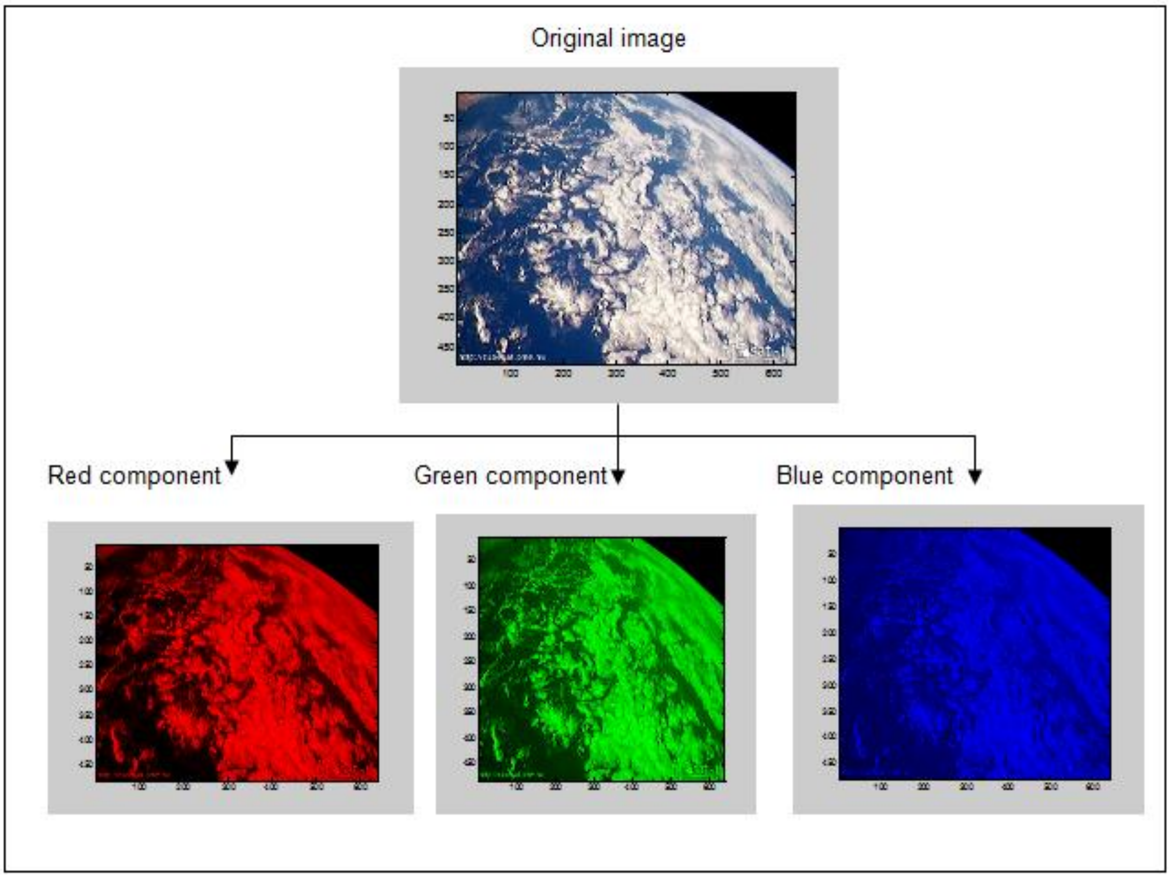


Figure 2.6 : The RGB image with its three components

(Adapted from Masat-1, 2012)

Categories of colour feature techniques include colour histograms, colour coherence, colour correlograms and colour moments. Colour sets are developed and utilized for colour image representation. Categories of colour feature techniques are illustrated in Figure 2.7.

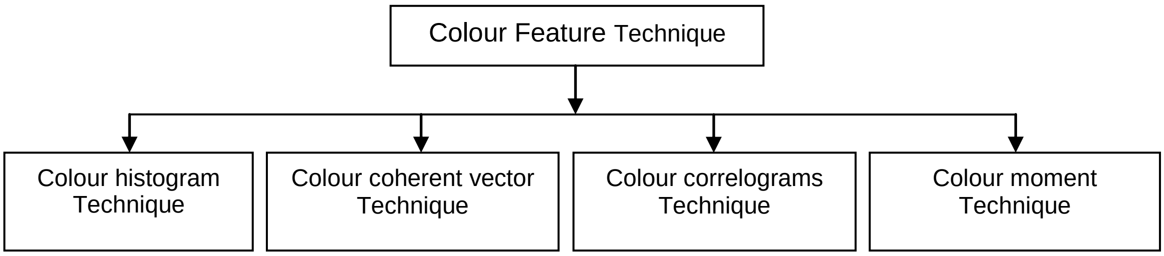


Figure 2.7: The categories of color feature

A. Colour histograms

A colour histogram uses a bar graph which depends on a colour space system to represent a particular colour in an image. The number of pixels embedded in images can be represented by bars, which are also known as bins. Two categories of colour histograms, namely Global Colour Histogram (GCH) and Local Colour Histogram (LCH), are identified. GCH takes the histogram of all images and computes the difference against a set of images using a similarity distance measure. The disadvantage is that it does not include information regarding every image region. In a colour histogram, it is hard to distinguish images that have the same quantities of the colour, even if the colour is in one area or extended across the image (Hussain, Rao & Praveen, 2013).

LCH is concerned with the colour distribution in the area of the image. It computes the colour histogram for each region separately and splits the image into fixed regions. The sum of each region's histogram represents the whole image. To compare two images using LCH, a distance is computed from one image to another image by computing difference in block of histogram of the images that are in the same location. When the LCH method is used, the efficiency of retrieving images is improved. But, this method is limited when the image is rotated or translated and takes additional time to compute (Fuertes, Lucena, Perez & Martinez, 2001).

As illustrated in Figure 2.8, the distribution of colour in an image is shown in a colour histogram. The x-axis of the colour histogram stands for the number of pixels and the y-axis of the bar stands for the amount of colour used in the image. When the same colours are found in different bars, the normalization method can be used to reduce the number of bars by putting them in the same bar. Normalization can decrease the content of information that may be gained from the image. The computation is efficient in a colour histogram and is insensitive to small changes in camera position. The limitation of a colour histogram is that it does not specify which colours are comprised in the image and their quantities (Wang, 2001).

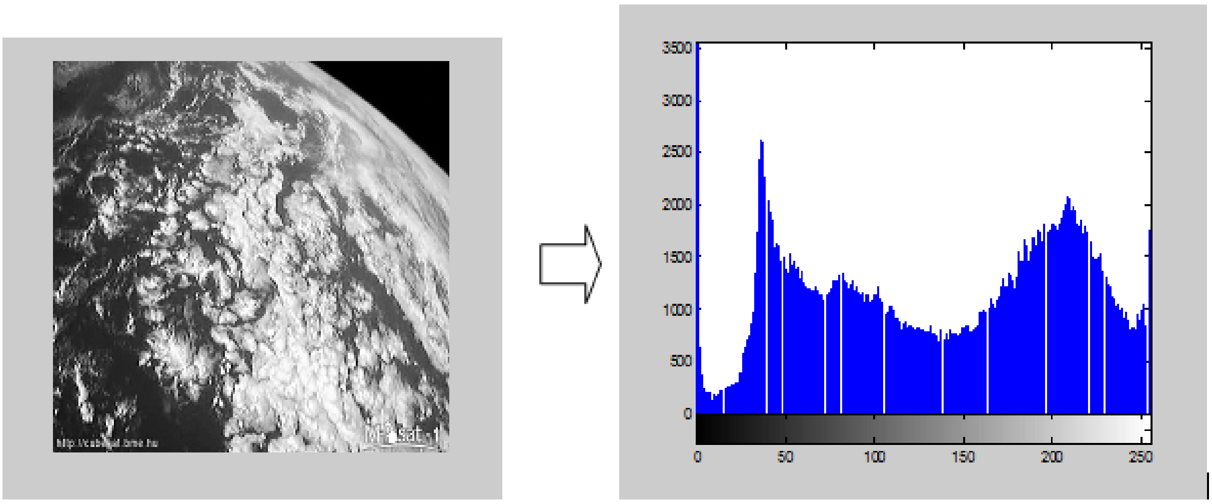


Figure 2.8 : Image and its histogram

B. Color coherence vector (CCV)

The CCV is classified into two groups, one with an incoherent vector and the other a coherent vector, both coming from a double colour histogram. Typically, a fixed threshold value of τ is used to determine whether an image's region is coherent or incoherent. When the region size value exceeds the threshold value of τ, the region is coherent whereas if the region size value is below the threshold value of τ, the region is incoherent (Kang, Yoon, Choi, Kim, Koo & Choi, 2007). The CCV overcomes the limitation of a colour histogram of identifying two different images containing the same quantity of colours to be the same. Yet, both images differ in appearance and objects comprised in them. The Colour Coherence Vector technique eliminates such confusion by determining whether one image's region colour is similar to other image's colour region using coherent and incoherent methods. For example, if the red pixels in an image are members of large red regions, this colour will have a high coherence, while if the red pixels are widely scattered it will have a low coherence (Pass, Zabih & Miller, 1996a). In an image, pixels of colour that are members of a big colour region are similar in coherence. Different images might contain the same amount of colours but differ from each other due to their colour coherences. Usually, CCV is used in image classification and it prevents two images having the same quantities to belong in the same group, as it ensures that coherence pixels or incoherent pixels of each image are unique. Thus, this technique allows fine distinction, unlike colour histograms confusing images having the same amount of colours. It performs much better than colour a histogram. According to Pass & Zabih (1996b), CCV bases its computation on a colour histogram, after the image is blurred to ensure that some pixels are changed by their average number in the region. Thereafter the colour space in an image is discretised in such a way that there are only *n* distinct colours in an image. The pixels in the stored array are then classified as either coherent or incoherent. Two pixels are compared and checked against their eight closest neighbours. The colour coherence region is obtained through computation of connected pixels within a given discretised colour array. Thus, an image is segmented according to the given colour space. Lastly, the largest eight colour coherent regions are chosen, which are enough to describe an image. Coherent regions are distinguished by counting which are coherent or incoherent (Hongli, De & Yong, 2004:762).

C. Colour correlograms (CC)

CC is a technique which enables the system to distinguish two or more objects in the image. It considers spatial correlation of colour and computes the pixel's distance value it compares. Colour Correlogram is widely used in image databases, where image content is used to query the database (Huang, Kumar, Mitra & Zhu, 2002). Colour correlograms are developed by quantizing an image's colour into m values c_i . . . c_m. The $k\varepsilon[d]$ distance values are determined, k is a constant and [d] is a set of distances between the pixels and $d\max$ is the maximum distance value calculated between the pixels. Using a table, each entry from the table (i, j, k) symbolises the probability of finding a pixel of colour c_i at a given distance from a pixel of colour c_i. To identify the image, colour auto correlograms, which considers colour pairs of the form (i, j), can be computed (Huang *et al.*, 2002).

Colour correlograms are widely used in image retrieval systems as it preserves the spatial correlations of colour information, and gives accurate image retrieval results. When compared to other techniques, such as colour histogram and colour coherence vector, it demonstrates high effectiveness (Taranto, Di Mauro & Ferilli, 2010; Zhao, Wang & Khan, 2011; Sirisha, Kumar, Vishnu & Srinivas, 2013; Murali, Raja & Bhanu, 2013; Tungakashthan & Intarasema, 2009). As stated by Talib, Mahmuddin & Husni (2013), the drawback of this technique lies in the expensive cost of memory space and computation time.

D. Colour moments

A colour moment depends on the probability distribution of colour in an image, and two or more images can be distinguished by computing the colour similarity between them. Usually, probability distributions are characterized by a number of unique moments, mean and variance values which distinguish their normal distribution. To identify an image, the probability distribution of colour should follow certain patterns, and the moment of that distribution can be used to identify an image based on its colour. As stated by Stricker and Orengo (1995), the image colour distribution can be computed using three colour moments, such as mean, standard deviation and skewness. Hence, these colour moments are defined:

Moment 1-Mean is average colour value:

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

Moment 2-Standard deviation is the square root of the variance of the distribution:

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

Moment 3 – Skewness is the level of assymmetry in the distribution:

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

Two images can be distinguished based on computed similarity between the distribution of colour and considering sum total of their weightage difference between the moments of their distributions. Formally this is:

$$d_{mom}(H, I) = \sum_{i=1}^{r} w_{i1} \left| E_i^1 - E_i^2 \right| + w_{i2} \left| \sigma_i^1 - \sigma_i^2 \right| + w_{i3} \left| s_i^1 - s_i^2 \right|$$

where (H, I): is the compared value of two image distribution,

i: the channel index,

r: the number of channels,

E_i^1, E_i^2: is the mean value of two image distributions,

σ_i^1, σ_i^2 : is the standard deviation of the two image distributions,

s_i^1, s_i^2 : is the skewness of the two image distributions,

w_i : is the weightage of each moment and d_{mom} value rank images, when d_{mom} value is small, the images are similar, whereas a larger d_{mom} value indicates that the images differ greatly (Stricker & Orengo, 1995). Colour moment applications includes colour image retrieval (Feng, Siu & Zhang, 2003).

2.5.2 Texture Feature

Texture is an important feature that represents the characteristic, pattern, composition and surface of an image. It has information on the structural arrangement of surfaces and their association to the environment (Srinivasan & Shobha, 2008). As shown in Figure 2.9, the texture feature is categorised into three categories, namely structural, statistical and spectral (Long, Zhang and Feng, 2002).

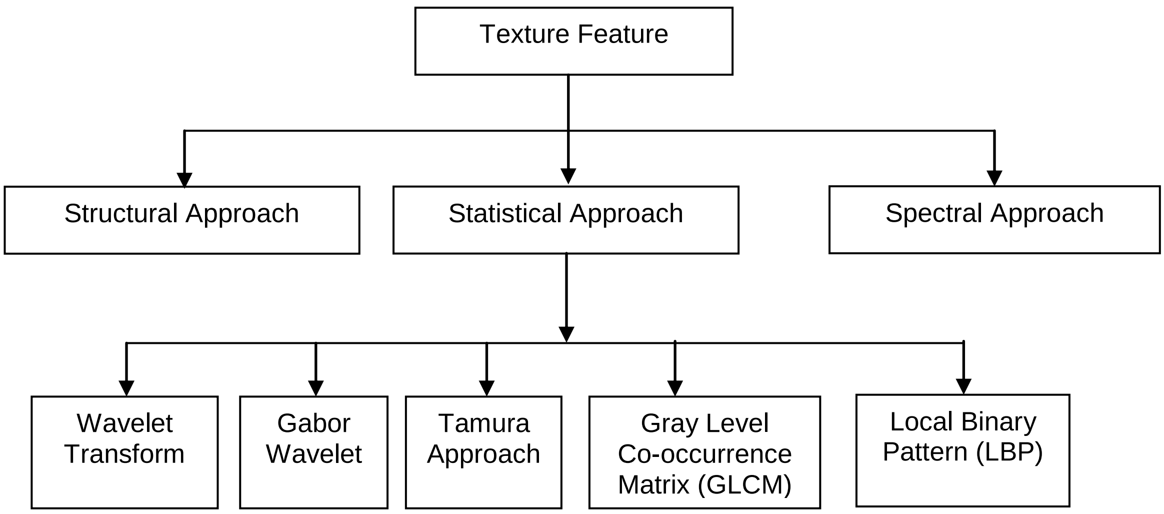


Figure 2.9: Block Diagram of texture categories

- *Structural approach* is texture which represents a micro-texture and macro-texture by defining the primitive and their placement rules. A particular location primitive and the probability of the chosen primitive can describe the image. The advantage of structural approach is that it can extract enough information to represent an object in an image. A useful tool to compute and analyse structural texture is provided by mathematical morphology (Hajek, Dezortova, Materka & Lerski, 2006).

- *Statistical approach* is a texture feature which is computed depending on the statistical distribution and intensity of a specific location relative to other intensity positions in the image. This approach includes Gabor wavelet, wavelet transform, gray level co-occurrence matrix and local binary pattern. The approach can be classified based on a number of pixels compared one pixel, two pixel and three or more pixels (Ojala & Pietikainen, 1999).

- *Spectral Approach* uses frequency domain to analyse texture contained in the image. It requires fourier transform to work on the original image to transform it into frequency space. Fourier transform does well by showing texture's strong periodicity and their performance deteriorates as the periodicity of texture weakens (Kale, Mehrotra & Manza, 2007).

A. Wavelet transform

A wavelet transform is a two dimensional image. By using wavelet coefficiency, useful texture feature information can be extracted from the image. The wavelet decomposes an image into four different scale levels or subbands. For example, when wavelet transform is used in a colour image, the four subbands (i.e., LL, HL, LH and HH) contain information to describe texture with a low resolution copy of the original image, and the other three regions are computed with three band pass filters in specific directions (Kumar and Esther, 2011:39). Each region contains content information which is useful texture features. Latha, Jinaga and Reddy (2007) proposed to use wavelet transforms for content based colour image retrieval. To determine retrieval performance of their system, they compared Haar wavelet transform, D4 wavelet and wavelet histogram schemes. In terms of retrieval rate, D4 wavelet transform outperformed both Haar wavelet transform and wavelet histogram.

B. Gabor wavelet

As stated by Ahmadian and Mostafa (2003) and Vacha (2010:11), the Gabor wavelet is a technique which is used to compute texture of an image. It is a simple and flexible method for capturing scale information and orientation of the image. It is based on function of two dimensions, having $g(x,y)$ function, which is defined as:

$$g(x,y)=\frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-\frac{1}{2}\left(\frac{x^2}{\sigma_{x^2}}+\frac{y^2}{\sigma_{y^2}}\right)+2\pi jWx\right)$$

$$G(u,v)=\exp\left\{-\frac{1}{2}\left[\frac{(u-W)^2}{\sigma_u^2}+\frac{v^2}{\sigma_u^2}\right]\right\}$$

where $\sigma_u = 1/(2\pi\sigma_x)$ and $\sigma_v = 1/(2\pi\sigma_y)$.

A non-orthogonal set can be created using the Gabor function and a signal is enlarged to give a concentrated frequency description. Gabor wavelets which is a class of self-similar function is computed by using $g(x,y)$ as the mother Gabor wavelet. A self-similar filter is found by using dilations and rotation of $g(x,y)$ by using the function:

$g_{mn}(x,y) = aG(x', y')\ \ a > 1, m, n =$ Integer

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

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

The a^{-m} is the scaling factor for energy which is dependant on m. The $\theta = \frac{n\pi}{k}$ and k are values of orientations. Al-Fayadh, Mohammed & Al-Shimsah (2012) concluded that when compared to another wavelet such as Haar, the Gabor offered accuracy more accurate result.

C. Tamura feature

Tamura features are texture features that can be extracted from an image, which include coarseness, contrast, directionality, line-likeness, regularity and roughness.

- *Coarseness:* This is an important feature for texture analysis. It deals with texture primitive and their size. It has small and large primitives, large primitives are small in numbers whereas small primitives are large in number in coarse texture (Lin, Chiu & Yang, 2002).
- *Contrast*: It is based on finding distinction in intensity between neighbouring pixels. It is also used as a measure of image quality. From persepective of texture, high contrast in an image gives a large value in intensity between the neighbouring pixels whereas low contrast gives a small difference (Lin *et al.,* 2002).
- *Directionality* is the region over which has global property. It gives orientation and shape to the texture primitive and its placement rule. Its orientation is more recognizable for primitives (Lin, Chiu & Yang, 2003).
- *Line-likeness*: It is line-like texture which is straight or wavelike primitives. It has only the shape of texture primitives, and its texture tends to give directionality (Mathur, 2012).
- *Regularity*: It is regular texture which has identical or similar primitives, and variations in their primitives. Regular texture has identical or similar primitives, and variations in their primitives, unlike irregular texture, which contains a variety of irregular or randomly arranged primitives (Chiu,Lin & Yang, 2001).
- *Roughness*: this refers to primitives and variation of physical surface. A rough texture has angular primitives, whereas a smooth texture has rounded, blurred primitives (Chiu *et al.,* 2001).

D. Gray level co-occurrence matrices (GLCM)

The GLCM was developed by Haralick *et al.,(*1973), who defined the following fourteen equations: Angular Second Moment, Contrast, Correlation, Sum of the Squares of Variance, Inverse Difference Moment, Sum average, Sum Variance, Sum Entropy, Entropy, Difference Variance, Different Entropy, Information Measures of Correlation 1,Information Measures of Correlation 2 and Maximum Correlation Coefficient (Haralick *et al,* 1973: 619; Gotlieb & Kreyszig, 1990; Partio, *et al.,* 2002; Liu & Liew, 2007; Feng, 2008).

It is a two dimensional, statistical texture feature of the second order which considers the relationship among two or a group of pixels in an image, and is widely used for feature extraction in content based retrieval systems. It uses gray level image to compute the relationship between pairs of pixels in an image, and the spatial relationship between two neighbouring pixels is determined in many ways by using offset and angles (i.e., $0^o,\ 45^o,\ 90^o and\ 135$). In one application, using a Matlab platform, a default is between one pixel and its immediate neighbour to the right (Pathak & Barooah, 2013).

E. Local Binary Pattern (LBP)

LBP is a texture technique which computes an image's pixels by converting binary value to decimal numbers, uses a centre pixel as a reference together with each neighbouring pixel and finally sums all binary values into a decimal number (see Figure 2.10). Its assigns a label to the pixel and compares the 3x3 neighbourhood of each pixel with the reference center pixel value, and summing the threshold values weighted by powers of two which results in a binary number. Hence, in this texture feature, a histogram to the value of 256 of different labels is used (Christiyana & Rajamani, 2012: 224-225; Sorensen, 2010:18). The value of the reference center pixel (a,b) of the image f(a,b) is computed by Equation 2.1.

$$LBP(a,b) = \sum_{i=0}^{7} U\left(f(a,b) - f(a,b)\right)2^t \qquad \textit{Equation 2.1}$$

Where $U(x)$ is the threshold function defined by Equation 2.2.

$$U(x)=\begin{cases}1 & if \quad x\geq 0\\ 0 & if \quad x<0\end{cases}$$ *Equation 2.2*

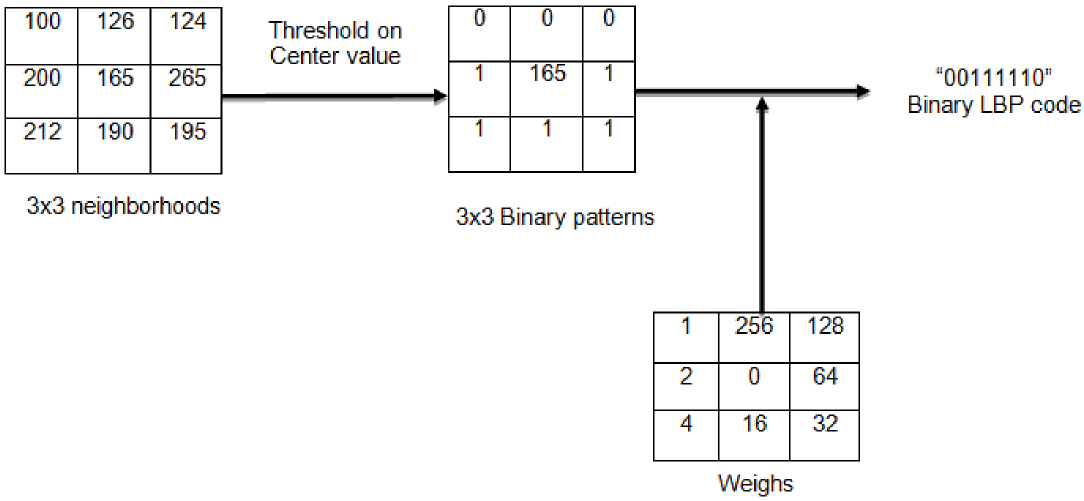


Figure 2.10: Basic LBP calculation

(Wang, 2013:18)

2.5.3 Shape Feature

Shape method comprises of two categories: boundary based and region based descriptors. Boundary based deals with the outline of an object contained in an image, while region based deals with the two dimensions or 'shape' of objects within the image. Boundary based is more widely used due to its simplicity, unlike its counterpart. Examples of boundary descriptors are chain code (Jahangeer & Baichoo, 2013), curvature and Fourier descriptors (Zhang & Lu, 2004).

A. Boundary based descriptor

The boundary based descriptors method requires extraction of boundary information which in some cases may not be available. The boundary technique includes chain code, moment invariants, Turning angles and Fourier descriptors.

- *Chain code descriptor:* Chain code uses four or eight connectivity to determine border pixels of an object in the image (see Figure 2.11). In each figure the line arrows indicate the length and direction, which are used to compute the object in the image by relying on either eight or four connectivity. Orientation of the objects is determined using angles (i.e., 45^o or 90^o), the chain code of an object is achieved by placing a grid on top of the image and the nearest grid nodes and boundary points are used to determine the codes. Numeration scheme method is used to determine the direction of each segmented code and through clockwise traversal of the boundary, shape descriptors values are collected. The chain code is sensitive to noise along the figure boundaries. This is overcome by superimposing another grid with a cell of increased size (Gonzalez & Woods, 2002).

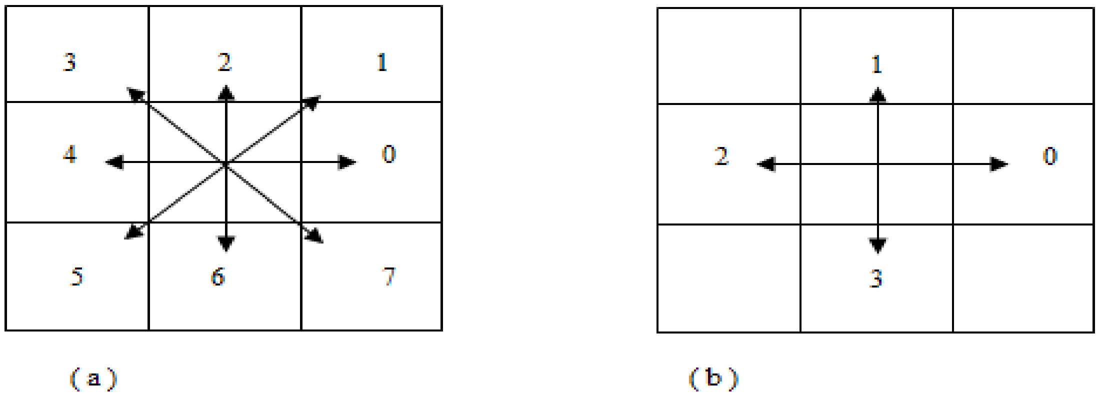


Figure 2.11: Chain code (a) Eight connectivity (b) Four connectivity

Chain code applications include fingerprint (Jiang, Zhao, Xu & Meng, 2009) and face recognition (Jahangeer & Baichoo, 2013; Seul,O'Gorman & Sammon, 2000).

- *Moment Invariants:* A series of properties which define shapes invariant to affine transformation within the image. They describe the properties of connected regions of the image (Li,Li, Fu & Yang, 2012).
- *Turning angles:* Turning angles describe a shape as a series of angles which define the angle between a tangent to a border pixel on the object and a predefined reference orientation such as the x-axis (Budd, 2007).
- *Fourier Descriptors:* Fourier descriptors are taken from the coefficients of the image represented in the Fourier domain by the Fourier transform (Budd, 2007).

B. Region-based descriptor

The region based method considers an image's region pixels to describe shapes of objects found in the image. Examples of region based descriptors are convex hull, moment descriptors, shape matrix and grid method.

- *Moment's invariant:*

  Moment invariant was developed by Hu (1962), and he depended on the algebraic invariants theory and derived seven invariants equations for allowing the rotation of two dimensional objects (Flusser, 2005). Moment invariants are classified as algebraic and geometric techniques. Examples of applications of the geometric technique are temple matching, registration of satellite images and character recognition. A drawback of the geometric moment's technique is that those derived from a few invariants from lower order moments are not enough to precisely describe a shape and high order moments are difficult to come by. Algebraic moments rely on the first central moment and predefined matrices of the eigenvalue, which are scale factors of the central moments. This technique works well on objects where the distributions of the pixels are found in the image and usually the outline of the shape is not important (Huang & Leng, 2010).

- *Grid based method:*

  This technique uses a grid to cover the entire shape of an object in the image. It starts computing the objects covered by the grid, starting from left to right and top to bottom. Each cell covered by a grid is assigned a value of "1" and cells not covered is given a value of "0". The 1-values of the covered cells are binary feature vectors. Distance measures, such as city block or binary hamming distance, are used to compute similarity between the two images' shapes. Typically, the shape of an object is normalized to allow the shape to be translated, scaled and rotated. A scale of fixed rectangular size is used and starts computing from the upper left corner of the rectangle and rotates the shape to a horizontal position. Flipped and mirrored shapes are usually done separately (Zhang, 2002).

- *Mathematical Morphology operator*

  Mathematical morphology was first introduced by Sera in 1982, to deal with digital image analysis problems. Its purpose is to use geometrical structures to extract information from an image. Mathematical morphology regards an image as a set and uses structuring elements to probe an image. It is based on Cantor's set theory and was initially used for binary images – improvements were made afterwards for grey level and colour images. The four basics operators of binary morphology are dilation ($\oplus$), erosion (Θ), open ($\circ$) and closed operation ($\bullet$). The four basics are described for image A and its structuring element B. As stated by Benoso and Nazuno (2008:109) the dilation operation of a set A by B, represented by $A \oplus B$ as Minkowski addition of A and B is more explained below:

  $$A \oplus B = \{(x, y) \mid B_{xy} \cap A \neq \phi\}$$ *Equation 2.3*

In Figure 2.12 the dilation process is shown whereby the image uses rectangular structuring elements in order to extract objects in an image.

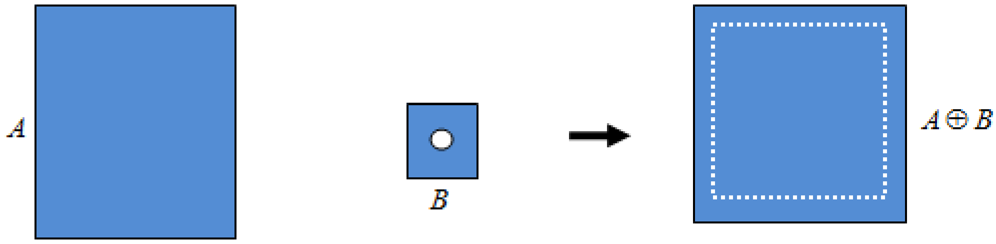


Figure 2.12: Dilation using rectangular structuring element
(Adapted from Karvonen, 2008:27)

As stated by Hongping and Yanping (2011:2291), and Fang and Yulei (2012), the erosion operation of a set A by B, represented by $A\Theta B$ as Minkowski subtraction of A and B is explained below:

$$A\Theta B = \{(x, y) \mid B_{xy} \subseteq A\} \qquad \textit{Equation 2.4}$$

Figure 2.13 details the erosion operation process whereby the image uses rectangular structuring elements in order to extract objects in an image.

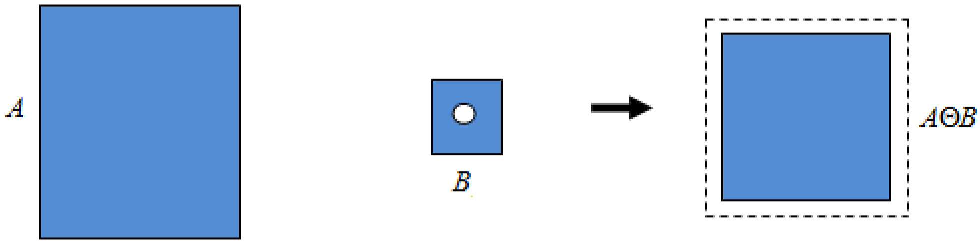


Figure 2.13 : Erosion operation using rectangular structuring element
(Adapted from Karvonen, 2008:28)

As stated by Hongping and Yanping (2011:2291), and Yu, Vladimirova and Sweeting (2008:1242), the erosion and dilation cause increases in the opening operation. Open operation is used with a given set and erosion is used after the dilation operation. Opening of set A and structuring element B are represented by $A \circ B$ which is explained below:

$$A \circ B = (A\Theta B) \oplus B \qquad \textit{Equation 2.5}$$

As illustrated in Figure 2.14, the opening operation eliminates thin protrusions from objects and creates a gap opening between objects connected by a thin bridge without reducing the size of the object.

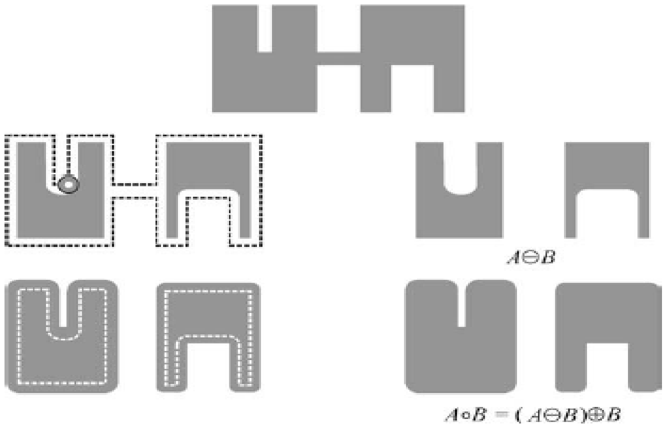


Figure 2.14 : Morphological Opening
(Adapted from Marques, 2011:311)

As stated by Benoso and Nazuno (2008:109), and Zhang, *et al*., (2010:895), the erosion and dilation cause increases in closing operation. Closing operation is used with a given set and erosion is used after dilation operation. The closing operation of set A and structuring element B are represented by $A \bullet B$ which is explained below:

$$A \bullet B = (A \oplus B) \Theta B \qquad \textit{Equation 2.6}$$

Figure 2.15 uses the closing operation to fill small holes and fuses narrow breaks where it closes thin gaps in the objects within an image, and the sizes of objects are maintained.

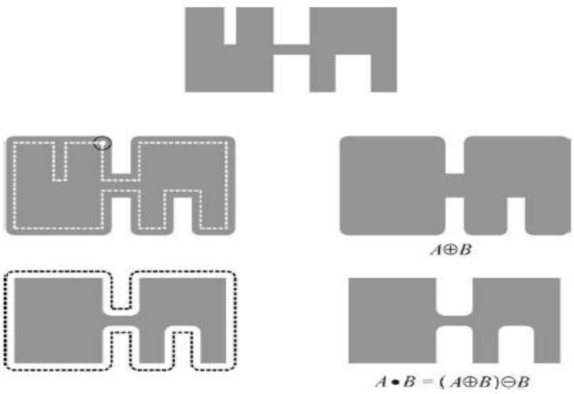


Figure 2.15 : Morphological Closing
(Adapted from Marques, 2011:312)

2.6 Image pre-processing

Image pre-processing, also known as filtration resolution enhancement, are techniques utilised to improve the quality of an image prior to processing into an application (Yong, Chixi, Bencheng & Zhi-Hao, 2013). Several image pre-processing techniques exist, such as mean or average filter (Rejeshwari & Sharmila, 2013:392), median filter (Chun-Yu, Shu-Fen & Ming, 2009), wiener filters (Marques, 2011:286; Bovik, 2000:131), discrete wavelet transforms (Ramdas, Keshav & Pradeep, 2012) and histogram equalization (Zare, Seng & Mueen, 2013:10).

Rejeshwari and Sharmila (2013) used image pre-processing on Magnetic Resonance Imaging (MRI) to improve the quality of an image so that the internal structures of the body tissue could be seen more clearly. They also compared different filters, such as average filter, median filter, wiener filter and discrete wavelet filter. When an image is denoised, the noise gets reduced by wiener filters and the resolution of an image is enhanced by interpolation based on discrete wavelet transform, which preserves the edges and contour information. To improve image quality, Rejeshwari and Sharmila (2013) remarked that both denoise and resolution enhancements are important for better performance.

Ahmed (2011) used image pre-processing in a car plate recognition system for Ethiopian number plates. A median filter was the choice for reducing noise and improving image quality. Hence, a colour image is changed into gray scale format and a median filter is utilized to eliminate noise and to improve the image's quality. To eliminate noise, a window of 3×3 median filters is used and improves the image quality. Other stages depend on this enhanced image to detect and extract the number plate.

2.7 Distance Measures

Most content-based systems use query by example to search similar images in the database by relying on distance measures (Castelli, 2001). A number of distance measures are identified, such as Mahalanobis distance (Chan, 2008:27), Euclidean distance (ElAlami, 2014:411; Sharma & Choudhary, 2013:61; Shan, Gong & McOwan, 2009:809), Chi-Square distance measure (Xianchuan & Qi, 2009), Histogram intersection distance (He, 2010:15), Manhattan distance (Lavoie & Merlo, 2012), Cosine distance (Dong-Cheng, Lan & Ling-Yan, 2007), Earth Mover distance (Shekar & Pilar, 2014,:220), Kullback-Leibler divergence (Log-Likelihood ratio) (Wells, 2007:40-41) and Quadratic distance (Edvardsen, 2006:20).

Qian, Sural and Pramanik (2002) compared two distance measures (Euclidean and Cosine angle distance) on small colour image databases. To compare the performance of different distance measures, precision and recall were used. Authors mentioned that selecting a suitable distance measure for an image retrieval system becomes a challenge. Choosing a distance measure must comply with how humans would recognize or find a similar image. When large databases are used, more computing resources are required for select distance measure. In their experiment, 650 colour images taken from a website were used. They concluded that both distance measures offered similar results when feature vectors were normalized by size.

As stated by He (2010), distance measures for histograms include Manhattan distance, Cosine distance, Normalized Histogram Intersection match, Quadratic distance, Earth Mover distance, Kullback-Leibler (KL) Divergence and Jensen-Shannon Divergence (JSD). Karthikeyan, Manikandaprabhu & Nithya (2014:512), reviewed distance measures which can be used for colour, shape and texture features. For the colour feature, their distance measures include histogram Euclidean distance, Minkowski metric, Manhattan distance and Canberra distance etc. For the texture feature, distance measures used include Earth Movers distance, weighted Euclidean distance, Kull Back-Leiber distance and Histogram Method etc. For the shape feature, the distance measures used are Angular Distance, Fourier Descriptor method and Polygon approximation method etc.

2.8 Summary and Conclusion

This chapter provided an overview of the CubeSat subsystems and also briefly touched on image compression: lossless and lossy compression techniques for both JPEG and JPEG 2000. The chapter also touched on feature extraction methods: firstly, colour feature methods such as colour histograms, colour coherence vector, colour correlograms and colour moments. Secondly, texture methods, namely wavelet transforms, Gabor wavelet, Tamura feature, Gray level co-occurence matrix and local binary pattern. Thirdly, shape features, namely boundary based and region based descriptors. Lastly, image pre-processing and distance measures were described. It was noted that shape feature categories included region based and contour based descriptors. Contour based were not suitable for complex shapes that consist of several disjoint regions i.e. trademark and logos, and are also generally sensitive to noise. The region based descriptors are more robust because they use all the shape information available in the image and provide more accurate retrieval. Table 2.2 compares the algorithms as presented within this chapter.

Table 2.2: Comparison of Feature extraction algorithms

Feature Extraction Algorithms		Advantages	Disadvantages
Colour (histogram)		• it is simple and the most often used colour feature • it is compact representation and low complexity • Represents the joint probability of the intensities of the three colour-channels (i.e., RGB). • It is efficient in retrieval system.	• It is sensitivity to quantization boundaries. • It is inefficiency in representing images with few dominant colours. • It lacks discriminatory power in retrieval of large image databases. • Does not match human perception very well.
Texture	Spatial texture	• Meaningful and easy to understand. • can be extracted from any shape without losing information	• Sensitive to noise and distortions
	Spectral texture	• Robust, • It need less computation	• No semantic meaning • need square image regions with sufficient size
Shape	contour-based method	• It is simple method • discriminate shapes mainly by their contour features • widely used in many applications	• It is sensitive to noise and variations. • It uses small part of information and in many cases, the shape contour is not available • shape content is more important than the contour features
	region-based method	• more robust as they use all the shape information available • provide more accurate retrieval • applied to general applications • can cope well with shape defection	• more complex than contour-based method

CHAPTER 3

SUPERVISED CLASSIFICATION

3.1 Introduction

"Image classification is when a number of images are classified into categories based on the availability of training data" (Dhasal, Shrivastava, Gupta & Kumar, 2012:123). Usually, each image feature, such as pixels, regions and lines, and edges elements, is considered for classifying images into categories or classes (Puzicha, Held, Ketterer, Buhmann & Fellner, 2000). Two main categories of image classification are identified, namely supervised classification and unsupervised classification (see Figure 3.1). The focus of this chapter, is to analyse and discuss supervised classification methods as put forward in Figure 3.2.

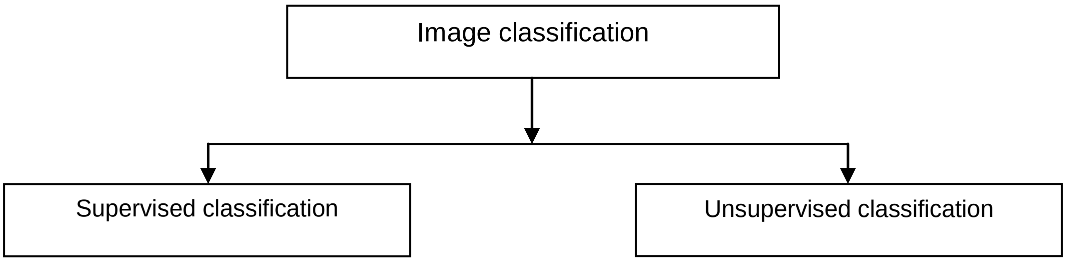


Figure 3.1: Image classification categories

According to Puzicha *et al.(2000),* supervised methods create a classifier that can precisely predict and assign new objects to certain classes. Although, experts are needed to label and categorize samples of known classes for a set of training data.

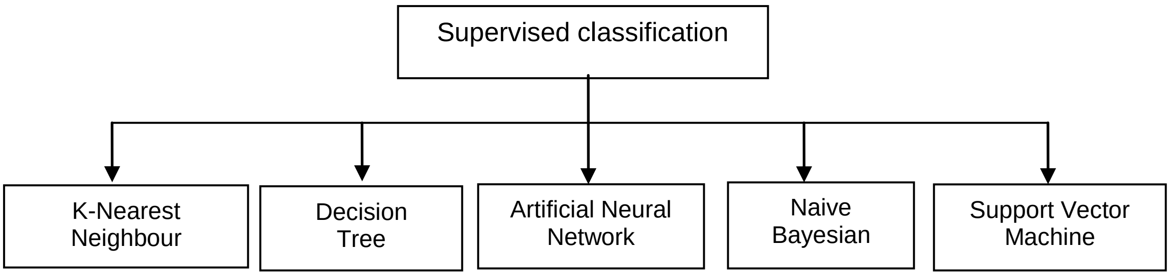


Figure 3.2: Supervised classification methods

3.2 K-Nearest Neighbour (kNN)

The kNN is the simplest classifier, and works well where prior knowledge of data is missing. It retains the entire training data set while learning and assigns new sampled data to the accurate class. It depends on two stages to function. In the first stage, it determines the nearest neighbour, often useful to consider more than one neighbour, and secondly determines the class using those neighbours (Abdelrahaman & Abdallah, 2013; Imandoust & Bolandraftar, 2013; Jivani, 2013).

When given a training sample and an unknown sample, the kNN classifier uses distance measures, such as Minkowski (Kurhe, Satonka & Khanale, 2011:2), Manhattan (Marques, 2011:486) and Euclidean (Narayana & Kulkani, 2012:57), to find where the unknown sample belongs. The computed values from a suitably chosen distance measure can determine which class of training data the unknown sample belongs to.

According to Aldayel (2013); Ramasundaram and Victor (2013); Kaghyan and Sarukhanyan (2012); Sugana and Thanushkodi (2010); Gejun Changesheng and Feng (2010); Geng, Liu and Qin(2008); Zhou and Chen (2006); Fuentealba and Charpentier (2004); Baoli, Shiwen and Qin (2003); Yang and Liu (1999), the kNN algorithm performed extremely well in experiments of different data sets. Table 3.1 lists the advantages and disadvantages of the k-nearest neighbour algorithm. When compared to other supervised classifiers, kNN achieves consistently high performance, without prior assumptions regarding the distribution from which the training examples are drawn.

Table 3.1: K-Nearest Neighbour (advantages and disadvantages)

(Bhatia &Vandana, 2010:303)

K-Nearest Neighbour	
Advantages	**Disadvantages**
Easy to modify for more complicated classification problems and can be used for applications in which an object can have classes' labels.	Choice of k : • When k value is too small, tends to be affected by noise and confuses the classes. • When too large, too many points from other classes are included, causing over-generalization. Good value can be given by cross validation method.
Outperforms other supervised methods such as Decision Trees, Naive Bayes Classifier, Support Vector Machine and Neural Network.	If there are two classes, an even number of k can cause a tie. Choosing an odd value of k prevents the ties.
It is easy to understand, implement and it perform well in different situations.	When dealing with small and large training samples of classes, most of the time k value selects large class of samples and tends dominate small samples of classes.
For a large number of samples it is nearly optimal	Large memory storage is required; all the results need to be stored until the algorithm completes the classifying.

3.3 Decision Tree

A Decision Tree is a tree structure with branches that represent outcomes of the test samples and leaf nodes that represent class allocations. It follows procedural rules to divide training samples by using fixed set of tests which are at each node. The generation of the Decision Tree depends on the given training sample. Thus, the developed Decision Tree is used to classify new data."Thus, developed decision tree is used to classify new data (Shen, Wu, Sun, Xiong, Fu & Xiao, 2011; Thakur, Markandaiah & Raj, 2010).

Examples of Decision Trees are ID3, C4.5 and CART. ID3 is the simplest technique to construct a decision tree by the principle of top-down approach, and the greedy method is used to test every node's given sets. Usually, a decision tree's nodes are tested from top node, root node and leaf node. Each node requires some tests to determine the level of leave nodes. ID3 is used in data mining classification of objects, network security and web attack detection and decision making purposes (Ming, Wenying & Xu, 2009; Hardikar, Shrivastava & Choudhary, 2012; Bhardwaj & Vatta, 2013).

The C4.5 originates from the ID3, a partition tree using gain ratio whereas CART uses the Gini coefficient with smallest values to test an attribute for a given set (Niuniu & Yuxun, 2010). There are broadly two types of decision trees: univariate decision trees and multivariate decision trees.

- *Univariate Decision*: input data of each node is of single feature. At each node data is split into two or more subsets from the input data of the single feature. Each test is required to have a discrete number of outcomes. This method is run by recursively dividing the input data until an optimal value of leaf node is reached and class value associated with the leaf node is assigned to the observation.
- *Multivariate Decision*: involves testing more than one attribute and are more accurate compared to univariate, but is very time consuming to generate and difficult to interpret (Witten & Frank, 2005). Multivariate trees are similar to univariate trees, except that each node test uses more than one input. It uses a linear discriminant function to estimate each node's attributes, and the coefficients of the linear decision functions at each node are estimated from the training data (Pal, 2002: 57-58).

Usually, decision trees tend to suffer from too many branches which is over-fitting of training data. To avoid over-fitting, some pruning methods were proposed, such as pre-pruning and post-pruning. In the pre-pruning method the growth of the tree is stopped before it reaches the point where it can perfectly classify the training data set. In post-pruning, decision trees are allowed to over-fit the data and is pruned after the tree is grown. Post-pruning is the most widely used method and is relatively simple to compute. An advantage of post-pruning is that it does not suffer from the horizon effect, which may cause inconsistency in the pre-pruning process.

The term Horizon effect is a problem that arises when a decision tree has to reach its optimal size of the final tree. An overly large tree causes over-fitting and gives bad results to new tested samples, and a small tree is unable to capture all the structural information in the sample space.The commonly used pruning techniques are Minimal Cost Complexity Pruning (MCCP) (Zhong, Georgiopoulos & Anagnostopoulos, 2008), Reduced Error Pruning (REP) (Mohamed, Salleh & Omar, 2012), Minimum Error Pruning (MEP) (Frank, 2000:63), Critical

Value Pruning (CVP) (Esposito, Malerba & Semeraro, 1997:480), Pessimistic Error Pruning (PEP)(Mahmood & Kuppa, 2012) and Error Based Pruning (EBP) (Wei, Wang, Yu, Gu, Wang & Yuan, 2009: 339; Elomaa and Rausu, 1999; Bruha, 2000). Table 3.2 shows the advantages and disadvantages of Decision Trees.

Table 3.2: Decision Trees (advantages and disadvantages)

Decision Trees	
Advantages	**Disadvantages**
Decision Trees are good for handling missing values whether of numerical and categorical input.	With large data sets, decision trees takes several hours to yield results and are computation-intensive.
Decision Trees' results are easily interpretable.	It suffers from over-fitting of training data, although pruning methods solves these problems
It can handle large data sets and usually give highly accurate results.	Most Decision Trees spend much time on growing partitions that will be pruned in the pruning phases; time wasting in growing of those partitions.
They outperform other algorithm interms in providing better accuracy. These other algorithms are artificial neural networks and support vector machine. High accuracy is maintained when data sets are increased.	It only works for small data sets and when used for larger data sets, the accuracy depends on selected features.

3.4 Artificial Neural Network (ANN)

As stated by Junfeng and Leping (2010:186), ANNs are mathematical models commonly used to predict performance in the systems. Its structure and how it functions is inspired from human biological neural networks (i.e. the brain). It imitates the human brain by memorizing and learning each time a task is given. Once trained on a specific task, it remembers it like the human brain does. It is a reliable technique and is composed of interconnected processing elements which function together to solve given problems. ANNs are widely used in many applications, such as the classification of offline handwritten signatures (Patil & Hegadi, 2014:6); MRI brain cancer classification (Jain, 2013); rainfall prediction (Nayak, Mahapatra & Mishra, 2013); Foliage-Plant identification (Kadir, Nugroho, Susanto & Santosa, 2011); digital image processing and classification (Siraj, Salahuddin & Yusof, 2010); image rating and classification of adults as non-adult images (Kim, Lee & Yoon, 2008); fingerprint Identification and recognition (Jin, Chekima, Dargham & Fan, 2002); face recognition (Jamil, Iqbal & Iqbal, 2001); classification of remotely sensed images (Kavzoglu, 2001); and to distinguish young corn plants from weeds (Yang, Prasher, Landry, Perret & Ramaswamy, 2000).

ANNs are used in complex situations to extract and predict future events. According to Moral (2004), neural networks are classified into two categories, namely feed-forward networks and feedback networks (see Figure 3.3).

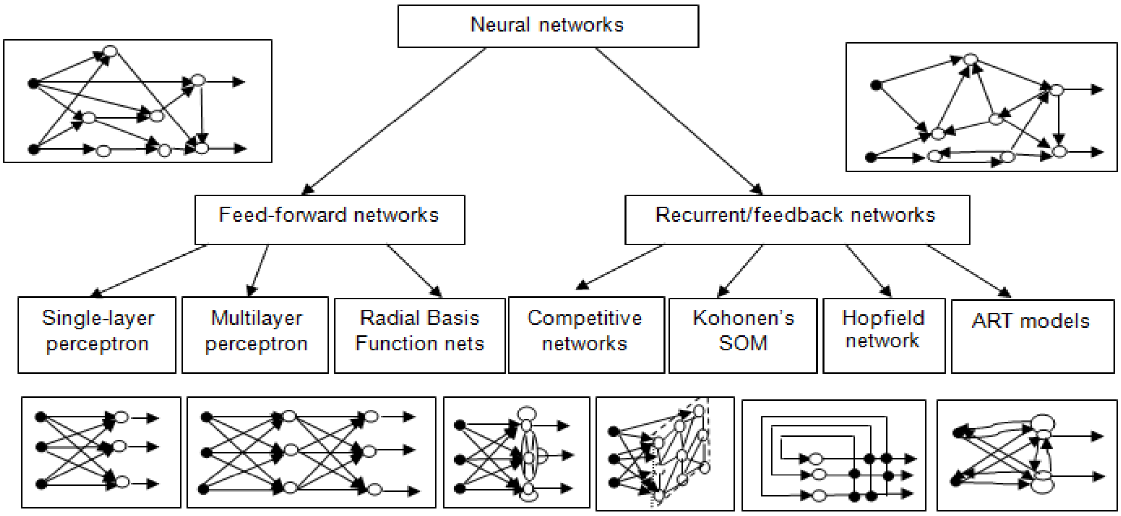


Figure 3.3: Taxonomy of feed-forward and recurrent network architecture
(Adapted from Moral, 2004)

3.4.1 Feedforward Networks

The Feedforward network is a fully interconnected layer that maps an n-dimensional input to an m-dimensional output. Feedforward is commonly used together with an error correction algorithm such as back propagation (Abdalla, Zakaria, Sulaiman & Ahmad, 2010), gradient descent (Rehman & Nawi, 2011) or conjugate gradient descent (Xiao-Shuai, Qing-Quan, Pei-Lin & Zhao-yang, 2010). Figure 3.4 shows a structure of the Feedforward network. Input layers are sets of circles, and inputs enter the hidden layer through the neuron weights. Each hidden layer has a sigmoid transfer function, and the output layer receives the hidden layer's output by a set of neuron weights. Inside each neuron in the output layer, there is a linear transfer function shown in the same figure, to provide the final results outputs (Bataineh, 2012).

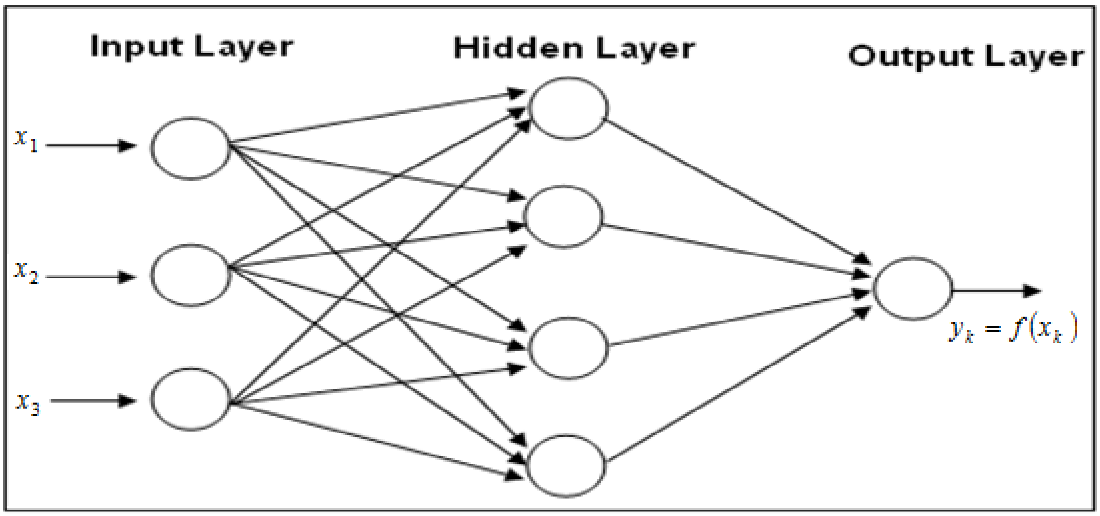


Figure 3.4: Structure of the feedforward network
(Adapted from Abdalla *et al.*, 2010:996)

3.4.2 Recurrent Networks

A recurrent network is a feedback fully connected network of closed loops. Each of its inputs has distinct input layers of nodes and each node can obtain input from every other node. They are widely used in problems such as learning a string of characters, wind turbine power estimation (Olaofe & Folly, 2012) and linguistics for language processing predictions (Tani, Nishimoto, Namikawa & Ito, 2008). The architecture of a fully interconnected recurrent network is shown in the Figure 3.5.

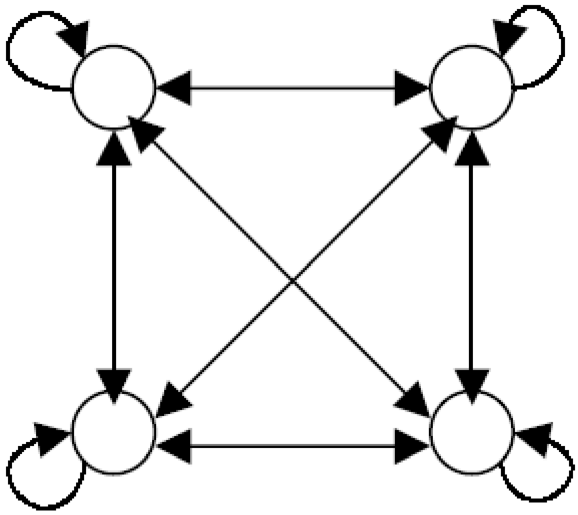


Figure 3.5: Fully connected recurrent neural network
(Adapted from Madsker & Jain, 2001)

3.4.3 Self-Organizing Map (SOM) Networks

As stated by Makarov (2012), Self Organizing Map networks (also known as the Kohonen network model) was introduced by Kohonen in 1980s. This technique requires an input data set to learn and form its own output representation for a problem. The basic idea behind SOM network is the formation of a 2-D array of interconnected neurons. When input data is fed to the network the response of each neuron is evaluated and the one which produces the maximum response, as well as those adjacent to it in the array, are modified so as to produce a stronger response to that input. After a number of presentations of each input pattern the system should ideally reach a state where an “ordered image” of the input is stored in the network. SOM represents a two layer structure (see Figure 3.6).

The first layer in the figure has a group of sensors which picks up the data passed to the network. This layer connects to the second layer which is referred to as the competitive layer, as it represents a regular lattice of neurons set in a specific topology. Competitive layer topology is determined depending on the number of neighbours that each neuron layer is directly connected to. The most frequent topology types are hexagonal and rectangular. In hexagonal, each neuron is connected to six neighbours in all directions and in rectangular, neighbours are only identified in horizontal and vertical directions,thus, resulting in a total of four neighbours for each of the neurons. SOM is mainly used in visualization of multi-dimensional data.

Specific applications of SOM network include Robotic mapping environments (Figueiredo, Botelho, Drews & Haffele, 2012); Fingerprint quality estimation (Makarov, 2012); Object replication (Soriano & Urano, 2011); Fingerprint classification (Turky & Ahmad, 2010); Analysis of genome signature strength of SARS coronavirus (Thamburaj & Ganapathy, 2010); Face recognition (Ghorpade & Agarwal, 2011; Deotale, Vaikole & Sawarkar, 2010; Monteiro, Queiroz, Carneiro, Souza & Barreto, 2006); Intrusion Detection System (Pachghare, Kulkarni & Nikam, 2009); Parallel Computing clusters (Liping & Wensheng, 2009); Case base reasosing retrieval (Hui, 2009); Urban Change Detection (Xue, Xiaowen & Jianwen, 2004); Chinese web page classification (Liang, 2003).

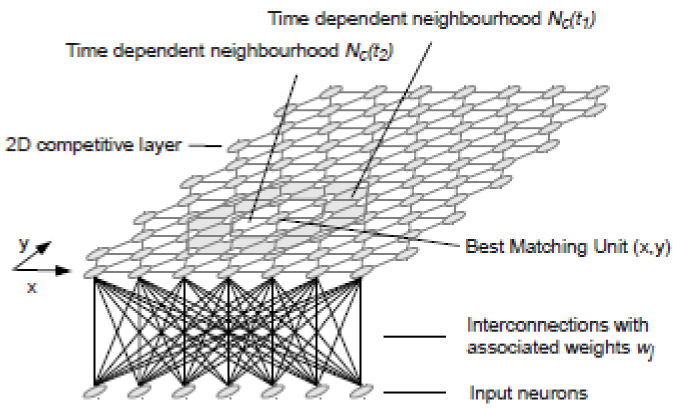


Figure 3.6: SOM Architecture
(Adapted from Makarov, 2012)

Table 3.3 lists advantages and disadvantages of Self Organizing Map(SOM) networks.

Table 3.3, shows Self Organizing Map (SOM)'s advantages and disadvantages

Advantages	Disadvantages
SOM networks are simple and easy to understand.	The number of clusters needs to be specified at the beginning.
SOM networks are well suited to parallel computation.	In order to classify, it requires necessary and sufficient data with no noise to provide accurate clusters.
It is a robust and efficient algorithm.	Using SOM requires a lot of time to train and is difficult in perfecting mapping where groupings are unique within the map.
Low computational complexity makes this algorithm very attractive for classification.	Its performance relies on the neuron's number. The more features in the vectors, the more inputs are needed. Due to this, SOM's behaviour and time of analysis is longer.
It can be used for classification without any preprocessing stage.	Computation intensive when used in image compression for digital images.
SOM is efficient in handling large datasets and robust when data set is noisy	Requires a number of neurons when data is increased. Thus, performance of algorithm is computation intensive and needs a lot of time to provide results.

3.5 Naive Bayes Classifier (NBC)

The Naive Bayes Classifier is a technique that uses Bayes theorem, which considers every feature as a class-condition independent. It is used in image classification of an object and works well with large data sets (Han, Kamber & Pei, 2012; Ferreira, 2010).

As stated by Kotsiantis (2007), When an NBC classifier is compared to a supervised classifier and Decision Tree, it performs much better than both classifiers. When applied to large databases, they offer high accuracy and speed. It is an easier classifier to implement and offers a high level of accurate results (Devi & Murty, 2002).

3.6 Support Vector Machine (SVM)

SVMs are classifiers based on a kernel method, which is useful in solving complex classification problems in several diverse applications such as image classification (Dhasal *et al.,*2012; Prasad, Savitri & Krishna, 2011); content based image retrieval (Rao, Kumar & Mohan, 2010; Guan, Antani, long & Thoma, 2009; Teng, 2006; Hang, 2004; Hong, Tian & Huang, 2000); weed/corn seedling recognition (Wu & Wen, 2009); road extraction; and image segmentation (Song & Civco, 2004).

According to Durak (2011) SVM is currently the best classification technique available and is also the most widely implemented classification technique. As stated by Gualtieri and Cromp (1998), SVMs are very efficient classifiers, with high classification precision and good simplification capability in solving problems. It utilizes a hyperplane to classify or solve a particular problem. The training sample of an SVM keeps computing the position of the hyperplane, and approximates the value of the hyperplane to provide an optimal classification solution. Usually, SVMs are two class classifiers but they can be expanded to solve multiclass classification problems. A number of binary classifiers are combined to solve multiclass SVM problems. There are broadly two types of Multiclass SVM classifiers, namely One-Against-One (1A1) and One-Against-All (1AA) (Anthony, Gregg & Tshilidzi, 2007).

3.6.1 One-Against-All (1AA)

One-Against-All is the earliest technique widely used to solve multiclass SVM problems (Melgani & Bruzzone, 2004). It starts by partitioning a big class into two class cases, for example, when classifying satellite images containing a variety of classes such as water, vegetation and built up areas. The class results are affected by what it is compared against, for example, water class compared to non-water area class, vegetation class against built up areas and vegetation against non-vegetation areas.

3.6.2 One-Against-One (1A1)

This technique creates a pair from each class, by training it to classify the two classes and the number of SVMs used as a result of N(N-1)/2. Given a number of large classes, an SVM leads to a complex classification system. Therefore, max-win or majority voting scheme is widely used for this type of complex system. When applied to a test point, each categorization offers one vote to the succeeding group and the point is labelled with the group having the most votes. In this approach, a modification can be made to give the weight age value to the voting process (Xu, Yin & Lv, 2009). The drawback of the 1A1 approach is that it is more computationally intensive (Gualtieri and Cromp, 1998). Table 3.4 shows SVM advantages and disadvantages.

Table 3.4: Support Vector Machine (SVM)

Support Vector Machine	
Advantages	**Disadvantages**
Produces very accurate classifiers.	SVM are computationally expensive.
Less over-fitting and robust to noise.	Selecting a kernel function is vital for the kernel algorithm's success.Thus, much time is spent looking for the best kernel function for a certain task.
The SVMs are good for larger data sets.	Selection of the kernel function parameters and slack variables can be challenging.
SVMs gives high accuracy and unique solutions when compared to neural networks.	It required alot of memory and is difficult to train and test when given large data set tasks.

3.7 Supervised classification: Comparison and conclusion

The simplest approach of selecting an appropriate algorithm is to approximate the correctness of the algorithms on the problem. Combining two or more algorithms, thus instigating hybrid algorithms, can increase accuracy. Table 3.5 compares the algorithms as presented within this chapter.

The study of supervised classification techniques has shown that the k-nearest neighbour technique is a labour intensive classifier. It only allows for good performance when working with small data sets. The k-nearest neighbour outperforms other classifiers given the data sets are small. Its drawback is that it requires extensive memory and also becomes computationally impractical when large data sets are used. The Decision Trees classifier offers better performance on large data sets as compared to k-nearest neighbour classifier. It uses tree structured graphs of internal and external nodes to classify the data. Its drawback is that it suffers from over-fitting of the training data, meaning it wastes time growing the tree. Some techniques have been proposed to eliminate the over fitting problem, such as pre-pruning and post-pruning methods. The artificial neural networks are mathematical models which are trained to predict specific behaviour and to remember that behaviour in the future like a human brain does.

Naïve Bayes classifier was compared to other supervised classifiers and it was determined that the Naïve Bayes classifier's performance is better than Decision Trees as well neural network classifiers. When applied to large databases they also offer high accuracy and speed. It is a simple technique to implement that obtains good results in most of the cases. SVMs are classifiers which depend on kernel function and linearly inseparable cases to classify training data into classes. They two categories of SVMs, One-Against-One and One-Against-All which are most used for multiclass SVM problems.

Table 3.5: Comparison of supervised classification techniques

Techniques	Advantages	Disadvantages
K-Nearest Neighbour	• Simple and easy to learn. • Training is very fast. • Robust to noise training data. • Optimal for large number of samples. • Outperforms support vector machine, Naïve Bayes and Decision Trees.	• Biased by value of K. • Computationally complex. • Large memory storage is required as all results need to be stored until the algorithm completes the classification. • A supervised learning-lazy algorithm. • Easily fooled by irrelevant attributes.
Decision Trees	• There is no excess degradation in the performance, when a smaller number of features at each internal node is used. • It provides a high level of accurate results for large data sets.	• Overlap when the number of classes is large. • For large tree level, there is error accumulation. • Optimal Decision Trees tend to be hard when designing.
Artificial neural Networks	• They learn to recognize the pattern which exists in the data set. • Fault tolerance: built in redundancy or the capability to withstand component failures without crashing • Associative recall: ability to retrieve information instantaneously based on content and to make an intelligent guess if there is no exact match for the required information.	• They have an inability to explain the model. They are built in a useful way, and get better results but difficult to explain how the results were obtained. • Analyst has to spend time understanding the problem and outcomes that will be predicted. If the data is not a good representative of the problem, the neural network may not produce good results. • It is time consuming, due to the learning of the system by an analyst who specifies the behaviour of the model.
Naïve Bayes classifier	• Easy to learn, and implement. • It mostly provides accurate results.	• When relying on assumption of class conditional, does not help in terms of holding. • Naive classifier is not reliable in modelling.
Support vector Machine	• Produces accurate results • Less overfitting and robust to noise • In non-regularities of data, the SVM is a useful tool for insolvency analysis.	• SVM are expensive and run slowly • The inability of SVM to deal with non-static data (dynamic data) sequences.

CHAPTER 4

UNSUPERVISED CLASSIFICATION

4.1 Introduction

Clustering, also called unsupervised classification, is the partitioning of a data set into clusters. For an object to belong to a cluster it must share the same characteristics as those in the same cluster as well as dissimilar objects that belong to their their unique cluster (Rai & Singh, 2010:1).

The similar or dissimilar attributes of clusters are computed using distance measuring techniques, such as Euclidean measure, proximity matrix and Manhattan distance measures. Methods of clustering include: hierarchical and partitional clustering methods (Gupta, 2011). As shown in Figure 4.1, within each category a number of sub-categories exist, which are discussed within this chapter.

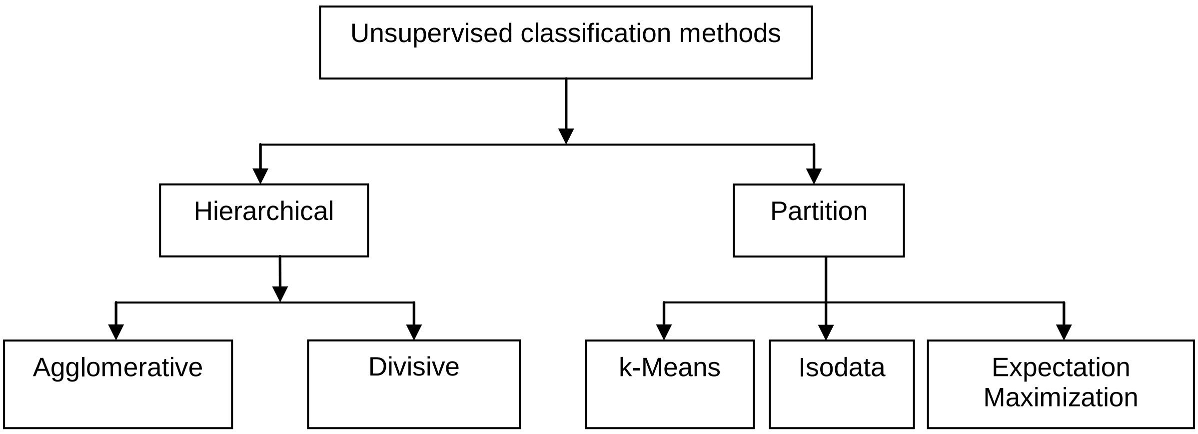


Figure 4.1: The categories of unsupervised classification method
(Adapted from Gupta, 2011)

4.1.1 Hierarchical Method

According to Aggarwal and Reddy (2014), hierarchical clustering produces a tree, also called a dendrogram, that shows the number of clusters. When the dendogram is constructed, the right number of clusters can be chosen by splitting the tree at different levels to get different clustering solutions for the same data, without running the clustering algorithm again. Hierarchical methods were created to overcome the drawbacks associated with partitional clustering methods. Two categories of hierarchical clustering, namely agglomerative and divisive clustering techniques, are identified. The agglomerative method merges clusters that are generated at the bottom levels, whereas the divisive technique separates clusters into small clusters. Although the divisive method splits and the agglomerative method merges clusters, both methods utilizes the dendogram when clustering the data and they differ only within the criterion used when clustering the data.

A. Agglomerative Hierarchical Clustering

This is a bottom-up technique, which starts by considering each object in its own cluster (relying on similarity or dissimilarity measures among a pair of objects) and then merging them into big clusters. This process continues until certain stop states are met (Han & Kamber, 2001; Yan, 2005; Purandare, 2004; Zhou, Zhang & Karypis, 2012). Examples of agglomerative methods include Single link (Mitsa, 2010; Aggarwal & Reddy, 2014), Complete link (Kantardzic, 2011; Aggarwal & Reddy, 2014) and Group average (Zheng & Xue, 2009; Webb & Copsey, 2011).

Applications of agglomerative hierarchical clustering include Fingerprinting (Lalhmingliana, Bhattacharyya & Sing, 2013), Web mining (Lee & Fu, 2008), Image database search (Rongjie, Jie, Pingjian, Fengjing & Guanfeng, 2008) and Music information retrieval (Li, *et al.,* 2012).

B. Divisive Method

According to Clarke, Fokoue and Zhang (2009:422), this method begins with all data being in one cluster which is then divided using distance measures until each subset remains with a single element. It works in reverse as that of agglomerative clustering which merges all the data points into one cluster. Examples of two divisive techniques are monothetic and polythetic. Monothetic is based on a single variable whereas polythetic is based on all variables participating in each division (Purandare, 2004). Applications of divisive clustering include automatic indentification refactoring software (Czibula & Czibula, 2008), Gene Expression (Kashef & Kamel, 2007) and document classification (Amuthajanaki & Jayalakshmi, 2013).

4.1.2 Partition Method

The objective of the partitioning method is to achieve a single partitioning group in order to collect every group existing in the data set. Objects that are similar are grouped together using distance measuring techniques. This method is commonly applied in remote sensing applications (i.e., classifying satellite images) and software packages that utilizes partitioning techniques such as k-Means, Isodata and fuzz k-mean. This is unlike the hierarchical cluster, which is uncommon in commercial remote sensing softwares due to being impractical when dealing with large data sets and computation complexity (Wilson, Boots & Millward, 2002; Larranaga & Lozano, 2002; Simpson, McIntire & Sienko, 2000).

A. k-Means Method

This is a simple, fast and easy iterative clustering technique that partitions a training data set into a number of clusters where initially the user provides values of k in advance (Sangita & Dhanamma, 2011;Wu & Kumar, 2009). According to Dabas and Chaudhary (2013), this method proves to be effective and produces good results. It is also widely used in document classification problems.

When k-Means is applied to artificial and real data, problems relating to the local minima may be encountered. In this regard different initial centroids are tried and the best results are chosen. Usually, its limitation is the difficulty in choosing the optimal value of k when working with a large data set (Wu & Kumar, 2009).

Examples of applications utilizing k-Means include pattern recognition (Gopi, 2007); image segmentation (Ng, Ong, Foong, Goh & Nowinski, 2006); text clustering (Xinwu, 2010); image halftoning (He & Zhan, 2011); data mining and object recognition (Yi & Moon, 2013); and analyzing recruitment data for hiring of employees (Sivaram & Ramar, 2010).

B. Iterative Self Organizing Data Analysis (ISODATA)

ISODATA is technique applied in clustering algorithms and belongs to unsupervised classification techniques. This technique splits clusters whereby the standard deviation is much larger than the given threshold. It follows an iterative procedure and needs the covariance matrices and class means for each class (Ahmad & Sufahani, 2012). Once the algorithm selects data, the natural grouping is known (Gu, Su & Du, 2004).

The steps of ISODATA clustering are: (i) enter the number of clusters, (ii) it randomly select cluster centres and subsequently allocate the pixels among the cluster centres, (iii) average the values of the pixels that are assigned to the class for the new cluster, (iv) compute new cluster covariance and mean, and categorize pixels to the closest clusters, (v) find differences between the original cluster and the new cluster. Steps (iv) and (v) are repeated when there is no satisfactory outcome in step (i) based on insufficient parameter values, or else the clustering procedure stops (Ahmad & Sufahani, 2012).

It has been implemented in several applications for example, as described by Liu, Lin, Cui and Dong (2007:1130), to classify extracted feature vectors of a palm print recognition system. Other applications include content based image retrieval (Banfi, 2000:94; Fang, 1997:16), VQ codebook design (Pan, 1996:111), land cover classification (Shi, Li, Yin, Fang & Song, 2010:3), deforestation (Eva, Carboni, Achard, Stach, Durieux & Faure, 2010:195), classifying land cover change and use in the Brazilian legal Amazon (Carreiras, Pereira, Campagnolo & Shimabukuro, 2006), infrared guided missiles detector (Rahimi, Shokouchi & Sadr, 2007), classifying remote sensing images and segmentation (Tarabalka, Benediktsson & Chanussot, 2009) and 3D segmentation of colour images (Atwan, 2012).

It allows cluster merging and splitting, and when a cluster has too many members it unites them to form a larger cluster. It uses a number of specified parameters to allow its execution to be adjusted for diverse sets of input feature vectors (Zhang, Fang, Liang, Wen & Wu, 2011). Its limitation is that it is slow in processing and classifying the data, mosty used where time is not a concern but accuracy of classifying data is an important issue (Wan, Wang & Song, 2012).

C. Expectation Maximization (EM)

The EM is an iterative technique used to approximate the parameters in the model and to construct the final combined result. When used where there is a lack of incomplete data it uses two steps, namely expectation step (E-step) and maximization step (M-step), to find a solution (Mladenovic, Porrat & Lutovac, 2011). Iterative EM algorithm is the preferred choice for finding the maximum posterior estimates of the unknown parameter. Every iteration process starts with searching for the optimal lower bound of the current estimated guess, and uses the current guess to maximize the bound to get a better estimate. Therefore, the current estimate and improved estimate depends on two steps, expectation step and maximization step (Dellaert, 2002). Usually, Expectation Maximization starts by assigning random values to all the parameters and alternates between the expectation step (current estimate is done) and maximization step (re-estimate parameters depend on previous estimate of current). These steps are repeated until the data converges (Dogdas & Akyokus, 2013). To produce accurate results, a model with estimate parameters is used (Yang & Blum, 2005). An advantage of the expectation maximization algorithm is that the convergence process is very smooth and insensitive to disturbances (He & Zhu, 2011).

4.2 Unsupervised Classification: Comparison and Conclusion

This chapter provided an overview of the two main clustering techniques, namely hierarchical and partition clustering, as well as their applications. In hierarchical clustering techniques, two categories, namely agglomerative and divisive, were explored in depth. As stated by Jin and Xiao (2013), the experiment with the agglomerative algorithm started with a single cluster at the outset, then successively merged all other pairs of clusters until all were merged into a single cluster containing all the data. If at the start, two data clusters are incorrectly merged, there is no way to rectify the error. Thus, errors will gradually accumulate or affect other pairs of clusters, leading to worse clusters. The divisive algorithm was found to be more efficient compared to agglomerative, as there is no need to generate a complete hierarchy all the way to the individual leaves. This algorithm uses big single clusters (of data objects) and partitions them into small clusters until each cluster has a single object. Partitioning clustering include k-Means, ISODATA and EM clustering algorithms. These techniques were found to be more robust and simple to implement as is shown in Table 4.1.

The k-Means, the oldest technique, is still used today after a number of improvements were made in order to make it faster and more robust. k-Means clustering divides a data set into a small number of clusters using selected k-values. EM clustering was found to be better in clustering missing data. Data containing noise gives inaccurate results and requires a large number of iterations to reach adequate convergence. EM clustering also does not take into account spatial correlation between the pixels in an image. Hierarchical clustering techniques groups the data objects in a flat partition and also arranges the data into a tree-like structure. Data objects are assigned to a leaf of the tree, while internal nodes represent groups of objects. For each pair of elements in a group, their distance is within a certain threshold. In the review of clustering techniques, it is found that partition techniques are more popular and more widely used than hierarchical techniques, and are found in a large number of software packages.

Chapter 5 that follows examines the requirements needed to design the image classification, storage and retrieval system.

Table 4.1: Comparison of clustering algorithm

	Type of clustering algorithm	Advantage	Disadvantage	Applications
Hierarchical algorithm	Agglomerative algorithm	• It can produce an ordering of the objects. • Smaller clusters are generated.	• Vagueness of termination criteria. • Not desirable for large data set applications.	• Classification of high resolution sensing images and segmentation. • Used in many biological studies.
	Divisive algorithm	• More efficient when compared to agglomerative.	• Computationally demanding.	• Gene expression. • Document analysis and classification.
Partition algorithm	k-Means algorithm	• Very simple to implement when solving practical problems. • Offer better quality results compared to hierarchical clustering.	• Very sensitive to initial starting values. • The results strongly depend on the initial guess of centroids.	• Pattern recognition and segmentation. • Image halftoning. • Data mining and object recognition.
	ISODATA algorithm	• Faster compared to k-Means algorithm • More adaptable and flexible than k-Means	• Run slowly, particularly with large data sets. • Has difficulty adjusting its parameters which control the convergence.	• Content based image retrieval. • Deforestation and land cover classification. • 3D segmentation of color images.
	Expectation Maximization	• Decreasing sensitivity to noise in an image. • Good in working with missing data.	• Requires a larger number of iterations to reach adequate convergence. • Convergence of EM algorithm is guaranteed only at a local minimum.	• 3D display where it is used for Isotropically Emissive displays. • Used in image segmentation and texture image classification.

CHAPTER 5

REQUIREMENTS ANALYSIS & DESIGN

5.1 Introduction

This chapter discusses software development process models such as the waterfall, incremental, prototype and spiral models. It also covers the requirement analysis and overview of the proposed design for the project.

5.2 Software Development Process

The image classification, storage and retrieval system for a 3U CubeSat requires the design and implementation of a software system. In order to build such a software system, an appropriate software development process model needs to be utilized as the basis for accomplishing the task of building the project. A number of software development process models are available in literature that are discussed within this section.

5.2.1 Waterfall model

The waterfall is a model that follows a sequential process for every stage. Within this model, software development is thought of as a sequence of stages where each stage proceeds from start to finish before the next stage commences. The output produced in one stage serves as input to the following stages and each stage is unique and solves different problems (Westfall, 2010).

Usually, the feasibility analysis is the first stage done and when feasibility of the project is achievable, gathering of requirement analysis and planning for the project can begin. Once requirement analyses are completed, design and coding are done, respectively. When coding is done and integrated in the system then the system testing process begins. When testing is a success, the system can be installed and users can operate the system (Jalote, 2008). The waterfall is suitable for applications where the software team has the knowledge to build a certain type of system. For example, when designing a web site and a new contract is given after to build a similar or same type of product, then the waterfall model can be utilized to accomplish the task (Futrell, Shafer & Shafer, 2002). The manufacturing and construction industries utilized the waterfall model extensively (Lehman & Sharma, 2011).

5.2.2 Incremental Model

The incremental is a model having several development stages taking place and can be termed a multi-waterfall cycle (Mahanti, Neogi & Bhattacherjee, 2012). It combines a module of the waterfall model in an iterative technique and all linear sequences create a deliverable increment of the software (Mujumdar, Masiwal & Chawan, 2012). The basic idea of this model is to build an increment by the same software making processes, such as analysis, coding and testing, according to customer requirements and afterwards increments of this are delivered to the customer. The first increment is known as the core product. As the customer uses the core product and is happy, the next development increment is planned (Sharma, Sharma & Mehta, 2012). It is the most popular model in several commercial software vendors and used in many prototyping softwares (Marciniak, 2001).

5.2.3 Prototyping Model

Prototyping is a technique which allows small modules to be created early in the software development stage (Marciniak, 2001). This model is useful when the requirements are known and a prototype is required. For example, a prototype model can be used in testing an incomplete version of the software program (Maheshwari & Jain, 2012). It focuses more on creating the actual software instead of concentrating on documentation (Sabale & Dani, 2012). The model begins with the requirements gathering, producing a rapid design according to the requirements. A model is prototyped, a customer evaluates it and a finally a product is produced (Sharma, Sharma & Mehta, 2012). There are two categories of prototyping models, namely Throwaway and Evolutionary.

A. Throwaway prototyping model

In the throwaway model method, a rapid and partial implementation of the system at some stage in or prior to the requirements stage is built. It is helpful in circumstances where requirements and the user's wishes are unclear or poorly specified (Mahanti, Neogi & Bhattacherjee, 2012). It is built rapidly for the purpose of showing it to the customer, to clear up any misunderstandings of the customer's needs at the beginning of the project. This model is for demonstration purposes and for the team to understand much better what needed to be done, not necessarly to the build overall system, and is discarded rather than becoming part of the final product. This is typically done quickly to demonstrate a proof of concept, and also to help a customer have a clear understanding of what the system will look like when it is complete. It is used to reduce requirement risks in the development stage of the product (Bidgoll, 2004:136; Kascheck, Kop, Steinberger & Fliedl, 2008: 509).

B. Evolutionary prototyping model

The evolutionary prototyping model is based on an evolving prototype that is not discarded. It is a prototype which is built based on some known requirements and understanding of the final solution. It is then refined and evolved instead of being discarded. Unlike the throwaway prototype which contains aspects of the system that are poorly understood, evolutionary prototypes are likely to be used with aspects of the system that are well understood and thus built on the development team's strengths. These prototypes are also based on prioritizing requirements in applications development (Kan, 2003: 21; Mahanti, Neogi & Bhattacherjee, 2012).

5.2.4 Spiral Model

The spiral model is a model that allows the merging of components of both partial developing stages and design, and other models use it to reduce risk in their product creations (Maheshwari & Jain, 2012). The spiral model is similar to the incremental model, but focuses more on risk analysis. It has four phases: planning, risk analysis, engineering and evaluation (Munassar & Govardhan, 2010). The spiral model is an iterative model that comprises risk analysis and risk management. Software is built upon a trial basis and verified if it accomplishes what the customer needs. If not, it is modified again to suit customer needs (Pressman, 2005; Sharma, Sharma & Mehta, 2012).

5.2.5 Comparison of software development models

Table 5.1 provides a comparison of the four software models.

Table 5.1: comparison of software development process

Types of model	Advantages	Disadvantages
Waterfall	▪ Simple and easy to understand and use. ▪ In this model phases are processed and completed one at a time. Phases do not overlap.	▪ No working software is produced until late during life cycle. ▪ Not a good model for complex and object oriented projects.
Incremental	▪ Easier to manage risks. ▪ Easier to test and debug when developing the modules	▪ Each phase of an iteration is rigid and do not overlap each other. ▪ Due to lack of gathering entire requirements of the software life cycle, problems may rise concering the system architecture.
Prototyping	▪ Cost effective. ▪ Increases system development speed.	▪ For large projects is not suitable.
Spiral	▪ Good for large & mission-critical projects.	▪ Costly model to use. ▪ Does not work well for smaller projects.

5.2.6 Selection of the model

F'SATI, as the customer, requires that a prototype system be built in a short period of time in order to ensure the final developed project falls within the developmental period of ZACUBE-02. In order to build such a system, the system engineers and development managers put forward initial requirement specifications and time-frames. In light of the fast development process required, a suitable testable prototype needed to be available for evaluation purposes and a low cost requirement. The software development model to be utilized in this project is the prototyping model.

The model selected guarantees that an initial prototype version of the system is built and presented to the customer in order to evaluate system operations and provide a clear view of what the expected system will look like when completed. This way of prototyping and showing a partial version at the start eliminates misunderstanding between the developer and client.

In this project, a prototype model under the category of an evolutionary model is to be used, reason being that as the customer outlines preliminary specifications, an involving prototype will be developed, shown and kept for future re-use. As new specifications arise, early versions will be revisited, re-used and modified in order to satisfy customer requirements in a very dynamic environment.

5.3 Requirements

The system requirements are the procedures that are followed, of what the system will accomplish and the services it can offer. In most of the cases, system requirements are based on what clients and users of the system want to see. There two types of requirements, namely user requirements and system requirements. User requirements are statements which detail services the user expects the system to provide and its limits. System requirements are descriptions which include detailed software functions, services and operations of the system, and documents are written giving information on how to implement the system. System requirements have two categories: functional requirements and non-functional requirements. Functional requirements are procedure details of what services the system will give, reactions of certain inputs and how the system will behave in certain conditions, and limitations of the system. Non-functional requirements are not concerned with aspects of how the system works, but on how to protect the system while is used i.e. security. After consultation with the F’SATI ZACUBE-02 design and development team, as well the project manager, the requirements of the system are as follows:

i. The functional requirements:

- Categorize satellite images as taken by the on-board ZACUBE-02 camera.
- Design, develop and maintain a database to be able to store a large number of images.
- Search the database using either extracted colour, texture or shape techniques.

- Identify similar stored images within a specifically designed database according to specified features (i.e. Colour, texture and shape) and criteria.
- Classify and store as well as retrieve images also according to time and date stamps.
- Retrieve image files as per identified qualities and features
- Display queried results for further analysis and use.
- Images received, manipulates and stored within the database will be in JPEG format.

ii. The non-functional requirements:

- The system developer should provide prototypes developed within a high level language to guide understanding.
- The system should calculate the features of a single received image in a reasonable time.
- The user interface should display the sample image used to search the database as well as the obtained searched results.
- The user interface should be user friendly and reliable.
- Security: Access to the database should be controlled by adequate authentication mechanisms.
- Availability: the system should allow authenticated users to continuously search for images within the database according to specifc criteria.
- Reliability and performance: the system should provide reliable image extraction according to user specificed criteria in a reasonable time frame.
- Maintainability: processess shall be in place to guide and ease maintenance and updates.
- Portability: the modules shall be portable to guide implementation on different platforms.
- The implemented system should provide accurate search results according to specified features.

5.4 Design

In this chapter a design for an image classification, storage and retrieval system for a 3U CubeSat is proposed as per the requirement analysis detailed in Section 5.3. Details regarding the design include a discussion on selected image classification techniques as well as feature extraction methods. The proposed design is highlighted in Figure 5.1 and shows the different components of which each plays a role within the functionality of the project.

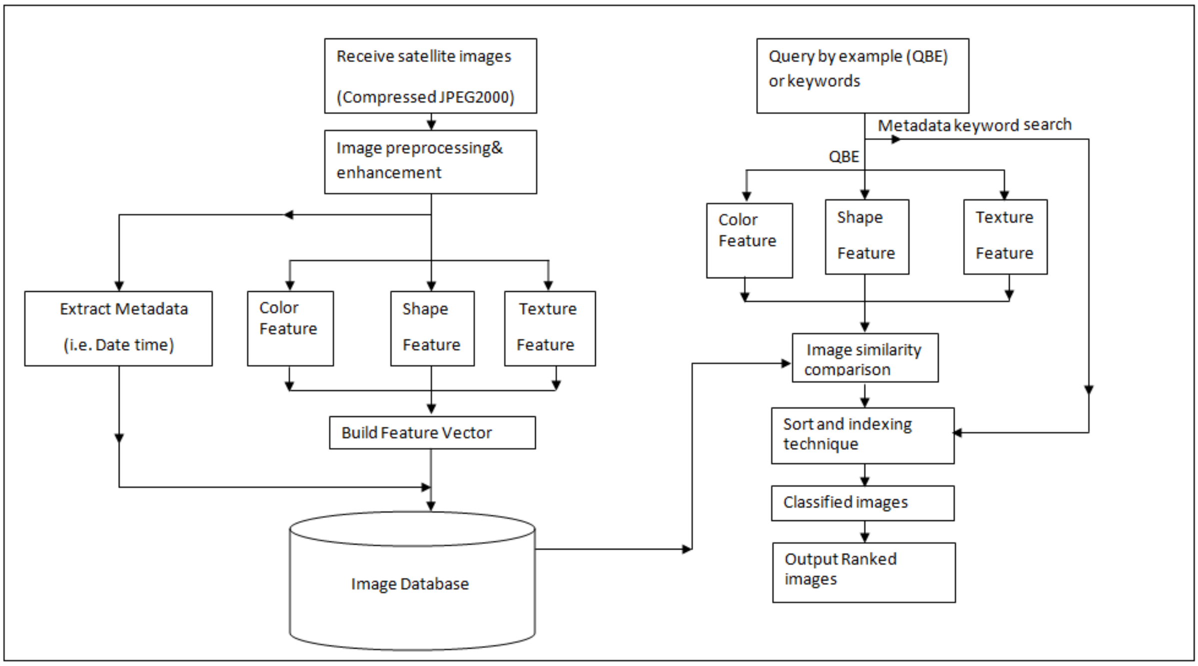


Figure 5.1: Proposed design

5.4.1 Receive satellite image

The orbiting satellite is equipped with an image compression system that reduces image size to allow the image to be downlinked to a ground station. Due to the reduced image file size, the onboard storage memory will temporarily store the captured images so that they can be transmitted to a ground station.

5.4.2 Image pre-processing and enhancement

Digital Satellite images are frequently distorted by noise due to errors produced in noisy sensor and communication channels. These errors either adjust pixel intensity while some of the pixels remain unchanged (Asubam, 2011). The proposed image pre-processing technique to be used to eliminate noise and improve quality of image is a median filter. In this filter, noise is removed in the image and the filter protects boundary information as it maintains image quality. The filter verifies every pixel against its neighbouring pixel to check if it represents the surrounding area. Thus, when the pixels are found that represent the surrounding area,the neighbouring pixels change accccording to median of individual values. The centre pixel is compared to the sorted pixel values of the surrounding neighbourhood to determine the median filter. Median is easily defined when its entries are odd numbers, and the brightness of each neighbouring pixel determines their arrangement (Chun-Yu, Shu-Fen & Ming, 2009).

5.4.3 Image metadata

Digital cameras are built with an internal clock which is used as a default when images are captured, and is embedded with the metadata of time and date of the internal clock of the camera (Sandnes, 2009). This image metadata stores information in different image formats such as Exchangeable Image File Format (EXIF), International Press Telecommunications Council's Information Interchange Model and Adobe Extensible Metadata Platform (XMP). As shown in Figure 5.2, the EXIF format provides useful information such as camera model, camera make, aperture time, EXIF version, exposure time, weight and height, x and y resolution, focal length and GPS coordinates including the time and date of when the image was taken (kakar & Sudha, 2012; Sundby, 2012; Gloe, 2012; Kee, Johnson & Farid, 2011).

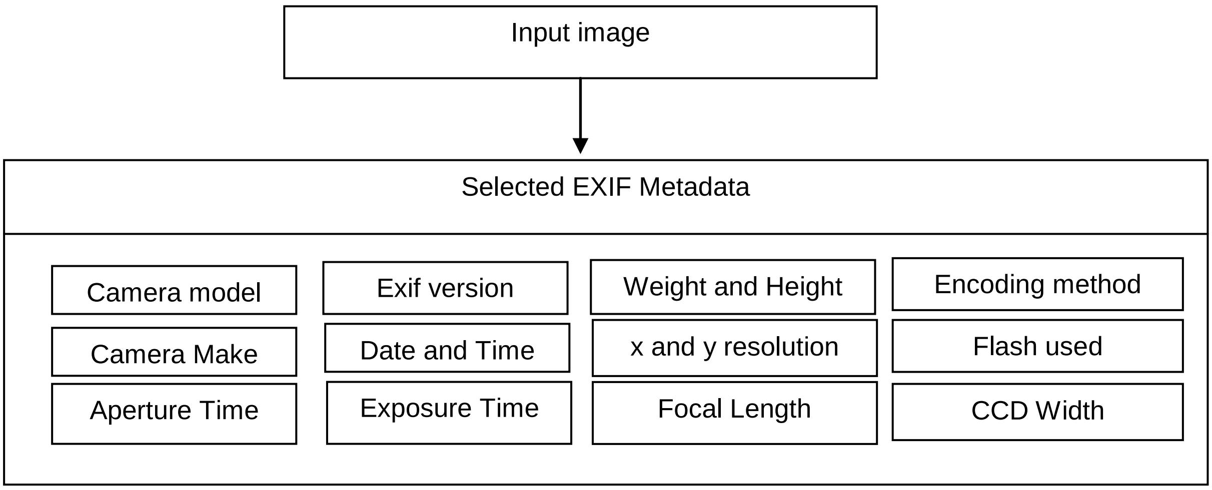


Figure 5.2: Image EXIF Metadata

The proposed image metadata extractor is detailled in Figure 5.3. When ZACUBE-02 satellite images are captured, they are then compressed and transmitted to the ground station. The received images are enhanced to remove noise and improve the quality of the images. Metadata information, such as date and time and image file name, are then extracted. The extraction of the metadata allows the users to retrieve the classified images according to the year, month and day on which it was captured.

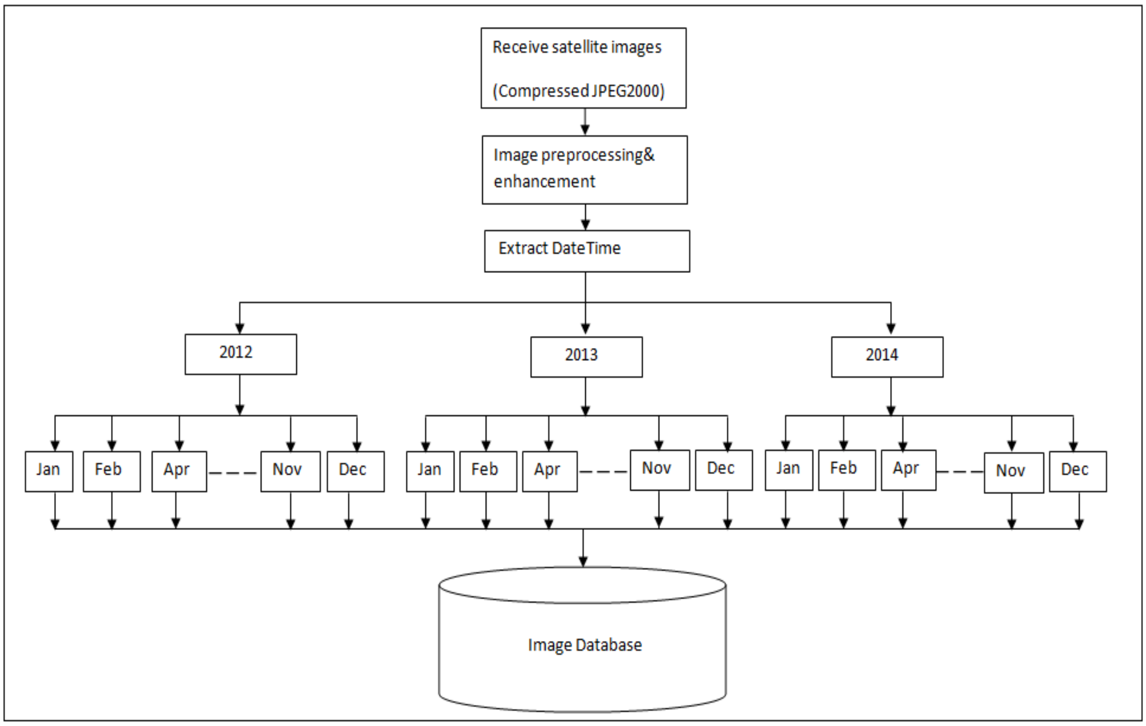


Figure 5.3: Block diagram of an image metadata extractor

5.4.3 Colour Feature

The technique used for colour feature extraction is the colour histogram, as it is a simple, accurate technique that offers better results compared to other techniques as discussed in Chapter 2. A colour histogram is a relative frequency occurrence of primitive colours in an image. It is a type of bar graph which represents the colour contained in an image. Depending on the size of the image, a number of bars are generated. Colour histograms have two categories, namely local colour histogram and global colour histogram.

As stated by Singha and Hemachandran (2012), a colour histogram for a given image is represented by the vector in equation 5.1:

$$H = \{H[0], H[1], H[2], H[3], \ldots \ \ldots \ \ldots, H[n]\} \qquad (5.1)$$

Equation 5.2, a normalization of the colour histogram, is provided. In order for images to be compared they must have same size dimension, otherwise the colour histogram has to be normalized. In the case of the CubeSat environment, only one camera will be used. Therefore, there will be no need to normalize images.

$$H' = \{H'[0], H'[1], H'[2], H'[3], \ldots \ \ldots \ \ldots, H'[n]\} \qquad (5.2)$$

$$H'[i] = \frac{H[i]}{p} \qquad (5.3)$$

Equation 5.3 is a colour bar histogram formula. $H'[i]$ represents the integer of pixels of colour i in the image and n is the total number of bars. In this project, the proposed colour histogram is used as shown in Figure 5.4, where the ZACUBE-02 satellite images are used as input to the proposed colour histogram algorithm. The algorithm includes a pre-processing technique to eliminate noise and improve the image quality. The histogram algorithm converts the original colour image to a gray scale image so that the histogram graph can be extracted from the image. The histogram is normalized to reduce image retrieval time. A connection to the database is then ensured so that the extracted histogram values can be stored in the database.

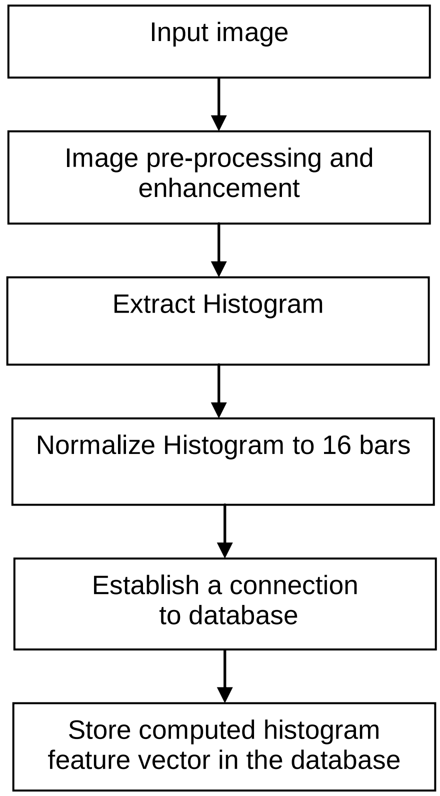


Figure 5.4: Block diagram of proposed Color Histogram method

5.4.5 Texture Feature

The technique used for texture feature is Gray level Co-occurrence matrix (GLCM) as it is the most commonly used choice for large database retrieval systems and it gives a high level of accuracy. This technique has proven to be popular in different fields, such as computer vision, pattern recognition and artificial neural network. The GLCM relies on gray scale images and pixels are manipulated to determine the texture occurrences. Therefore, when a colour image is transformed into a gray scale image so that GLCM can be computed, as it only depends on black and white pixels. Figure 5.5, shows a block diagram of the proposed GLCM texture feature. The input image is received from the satellite and is used as input to the system. GLCM values, such as Correlation, Homogeneity, Contrast, Mean and Energy, are computed. To determine the GLCM value, the colour images must be changed to gray scale images (the gray scale levels are distinguished with numbers ranging from 0 (black) to 255 (white).

When a gray scale image contains many gray levels, the more texture information is found, but also an increase in computational costs. Therefore, colour images are transformed to gray scale to eliminate the hue and saturation details and preserve the luminance. GLCM computation is done to provide a way to recognize similar images. These features are extracted and averaged over the four directions. All sets of features derived from the four directions are stored in the database to be used for search and retrieval. Contrast values are computed to enable distinguishing of the objects within the image, homogeneity is used to determine distribution of elements in diagonal of GLCM and energy gives the amount of squared elements in the GLCM (Munshi, 2010).

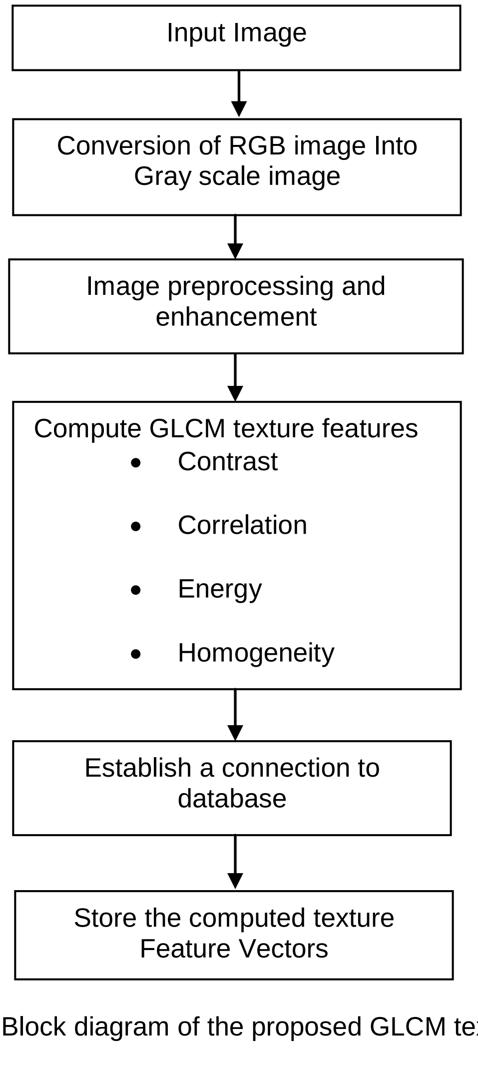


Figure 5.5: Block diagram of the proposed GLCM texture Feature

5.4.6 Shape Feature

The technique used for shape feature extraction is Mathematical Morphology. Mathematical Morphology is a region based shape feature extractor. It has been used for pre and post processing of images containing shapes of interest in many applications. It depends on the design of structuring elements, their size and shape which is important to the success of the morphological algorithm/process that uses them (Marques, 2011).

Mathematical morphology works from the objects within an image, using a structuring element of specific size, and shapes of the objects. The unwanted protruding areas on objects can be eliminated by using dilation (adding pixels to the boundaries of objects) and erosion (removes pixels from boundaries of objects). However, the number of pixels removed or added to the objects depends on the size and shape of the structuring element. Figure 5.6 shows the block diagram of the proposed mathematical morphology shape feature. In the figure, the input image is transformed from colour to gray scale, and pre-processed to remove noise and improve the quality of the image. The object's three aspects feature values, namely area, roundness and perimeter, will be extracted. The database connection is established to allow the extracted features to be stored within the database.

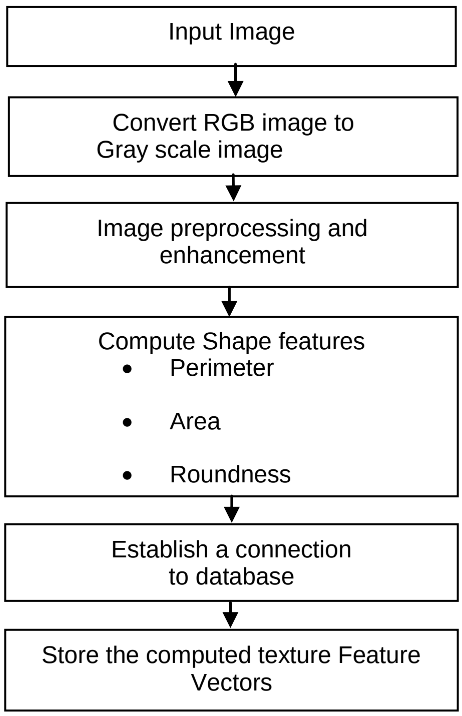


Figure 5.6: Block diagram of mathematical morphology shape feature

5.4.7 Image Database Model

A database management system is able to create, search, query and retrieve data from a database, provided a user interface does exist or is developed. A relational model stores data in the form of a table with rows and columns, and a foreign key can link two different tables' information. The database model is designed as shown in the Figure 5.7. It will be able to store and handle the extracted features. It is important that the developer of the system creates enough tables with their appropriate data fields, and data stored should be easily retrieved. In this project, features vectors such as colour, texture, shape and metadata should be stored in the database.

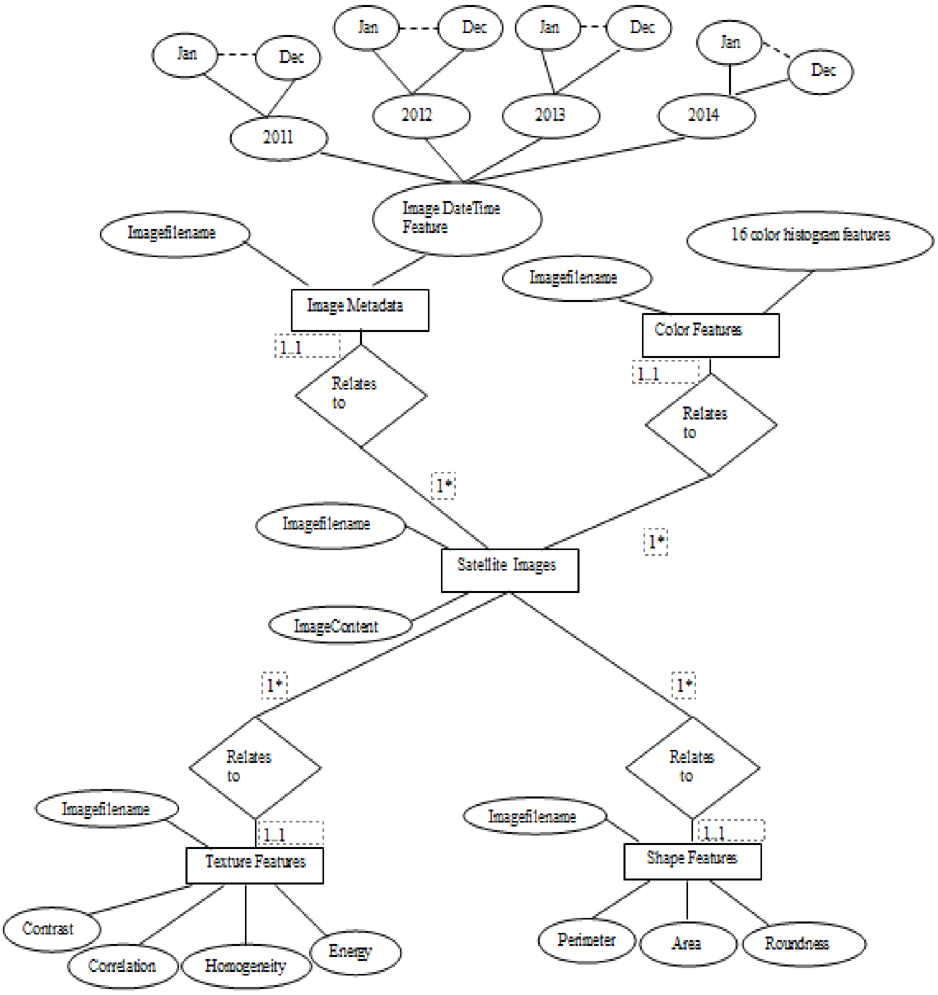


Figure 5.7: Entity relationship data model diagram for the system

5.5 Conclusion

This chapter has been concerned with the software development process, requirement analysis and the proposed design with its components. The software development process models, namely the waterfall model, incremental model, proptotyping model and spiral, were discussed. The requirement analysis, including both the functional and non-functional requirements needed to design and develop the project, was discussed. The proposed design components for extracting features in an image were image metadata, colour features, shape features and texture features. In the colour feature, a colour histogram method was proposed. For the shape feature, mathematical morphology was proposed. For the texture feature a gray level co-occurrence matrix (GLCM) method was used. An overview of the design implementation, proof of concept, and hardware and software needed to implement the system are provided within the next chapter.

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

This chapter describes the hardware and software used in the implementation process of the image classification, storage and retrieval system for the 3U CubeSat. The implementation focuses on the proposed design methods as well as the testing tools to test functionality of each extraction method. Figure 6.1 details how the implementation as a proof of concept will be built. A Wang database containing one thousands images, having ten groups, each group consisting of one hundred images will be used to verify and test all algorithms proposed. In the figure group 1 up to group 10 will each contain one hundred images to prove algorithm effectiveness. The accuracy of each algorithm , will be tested on sets of 14, 25, 50 and 100 images per group.

Figure 6.1: Block diagram of the implementation of the algorithms

6.2 Hardware setup for the system

Table 6.1 provides an overview of the specific hardware and software used as implementation platform. Software includes Matlab and MySQL on a personal computer running the Windows 7 operating system. Hardware requirement specifications are an Intel core i5-520M Processor, 3.33 GHz, 4GB of memory (RAM) and 500GB of hard disk drive.

Table 6.1: Hardware and Software platform

Type of software	Version
- Matlab platform	- R2009b
- MySQL Database	- 5.5
- Personal Computer and hardware	- Running Windows 7 - Intel core i5-520M processor - 3.33GHz,4GB of Memory(RAM) - 500GB of Hard Disk Drive

The MATLAB platform includes toolboxes used to allow the Graphical User Interface (GUI) to interact with users and also built-in image processing functions which are valuable in extracting features in an image.

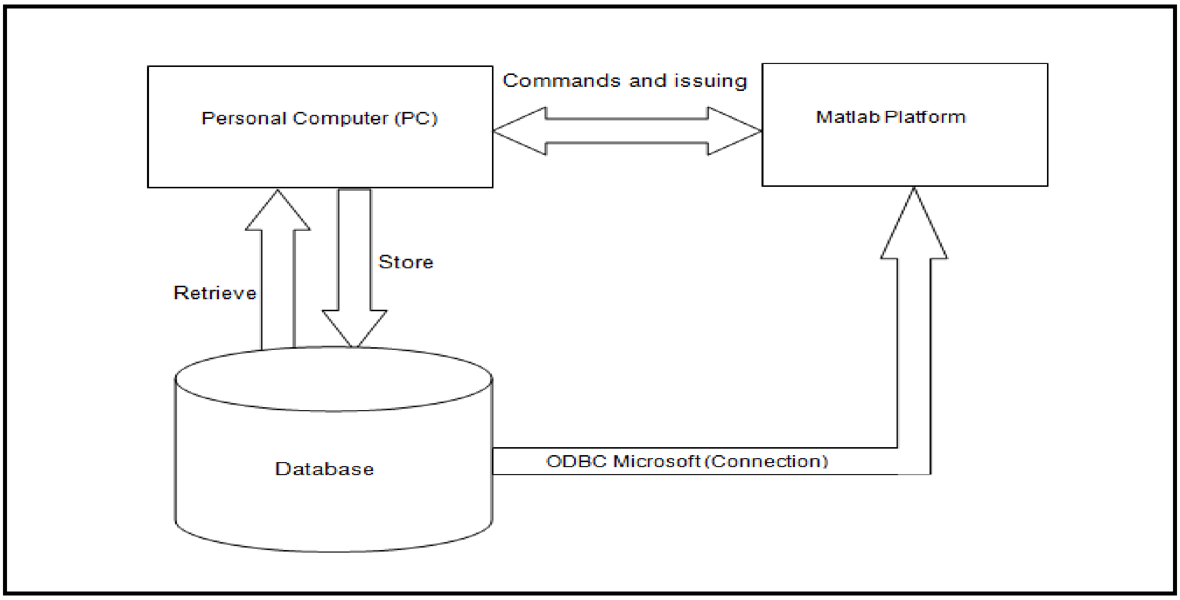


Figure 6.2: Experimental set-up

The database used is MySQL 5.5 as it is open source and efficient in storing and retrieving data. It also interfaces with software platforms such as MATLAB (see Figure 6.2). Communication between MATLAB-MySQL database is achieved using ODBC Microsoft and a MySQL driver. The installation and interfacing details regarding the MATLAB-MySQL Database are available in (Appendix C and D).

6.3 Graphical User Interfaces(GUIs)

MATLAB has two ways of building graphical user interfaces. In the first method, a GUI is built using the Graphical User Interface Development Environment (GUIDE) and the second method is by using programming code. Within GUIDE, a design is being populated with various GUIDE library components placed onto the graphic layout editor template. The GUIDE creates associate code files containing call-back functions for each component. GUIDE saves both the figure (Fig-File) and code file. The code is only generated once a figure file is saved by default name or changed to a suitable name associated with the project. In the second method, a GUI is developed using programming code. The codes are created to provide the look and feel of the GUI. When a code is executed, a file is created and contains all populated components. Unlike GUIDE, in programming code the developer doesn't have to save between sessions when developing the code. The choice of method depends on the experience of the developer preferences and the kind of application needed to produce the GUI. In this project, both techniques are utilized.

6.3.1 Colour Feature GUI and extraction method Development

Figure 6.3 shows the GUI template that contains layout editors which has horizontal and veritical grid lines to allow users to guide the components to their proper positions. The figure also shows the GUIDE library components, namely Push buttons, Radio button, Toggle buttons, Check Box, Edit Text, Table, List Box, Pop-Up Menu, Button Group, ActiveX Control, Slider, Menu Item, Static Text and Axes, that may be utilized. For more detail regarding each GUIDE library component, please refer to Appendix E.

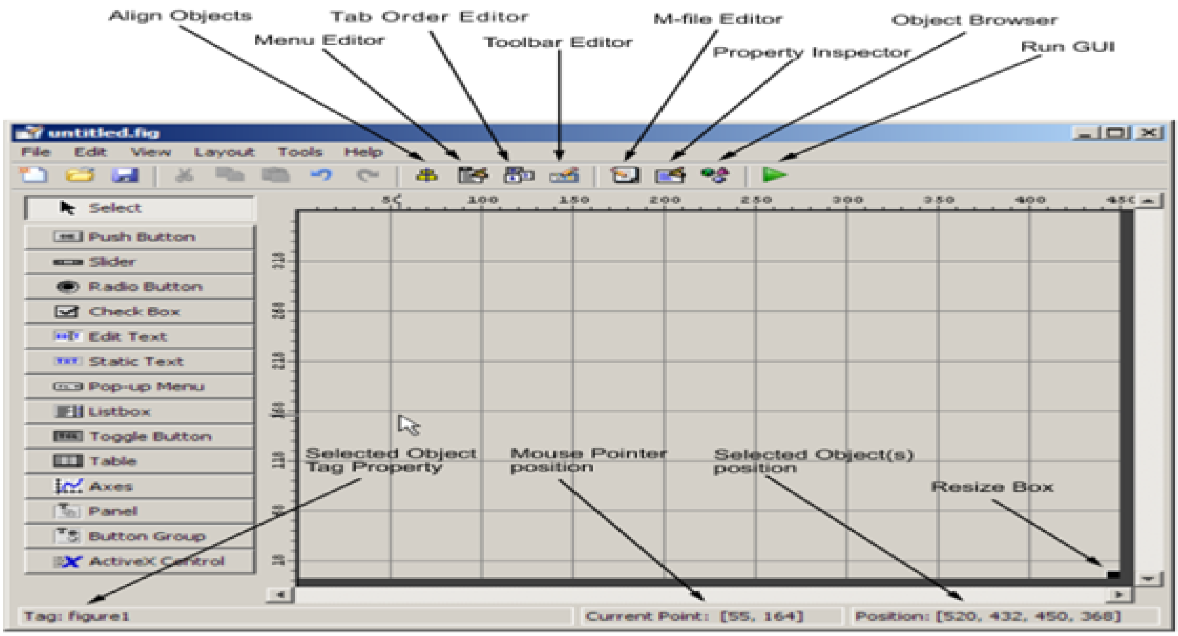


Figure 6.3: The new GUI template

Each button component is usually associated with a call-back function, to ensure that when a button is clicked or toggled, the call-back is triggered. The call-back function is event driven, that triggers execution on input received. If a call-back is executing and another event is triggered for a call-back, it will attempt to interrupt the call-back that is already executing.

Figure 6.4 shows the GUI for the colour feature that consists of three main component panels:

(i) Colour feature panel

(ii) Select query and view panel

(iii) Display result panel

The colour feature panel includes edit text, slider and measure weightage buttons. The purpose of 'edit text' is to display the value of the slider, ranging from 0 to 1 of the weightage of the feature. The measure weightage combines the slider's value with the extracted value of the query image so that a search can be done against images in the database. Select query and view panel displays the choice of the query image to be used against images in the database. It contains two buttons, one to select a query and a second to search the database. The display results panel consists of twelve axes components which are used to display the retrieved results which are similar to the query image. The associated source code of the colour feature GUI developed is available in Appendix A: Listing A.1. Sections 6.3.2, 6.3.3, and 6.3.4 follows a similar layout of the components panel discussed above.

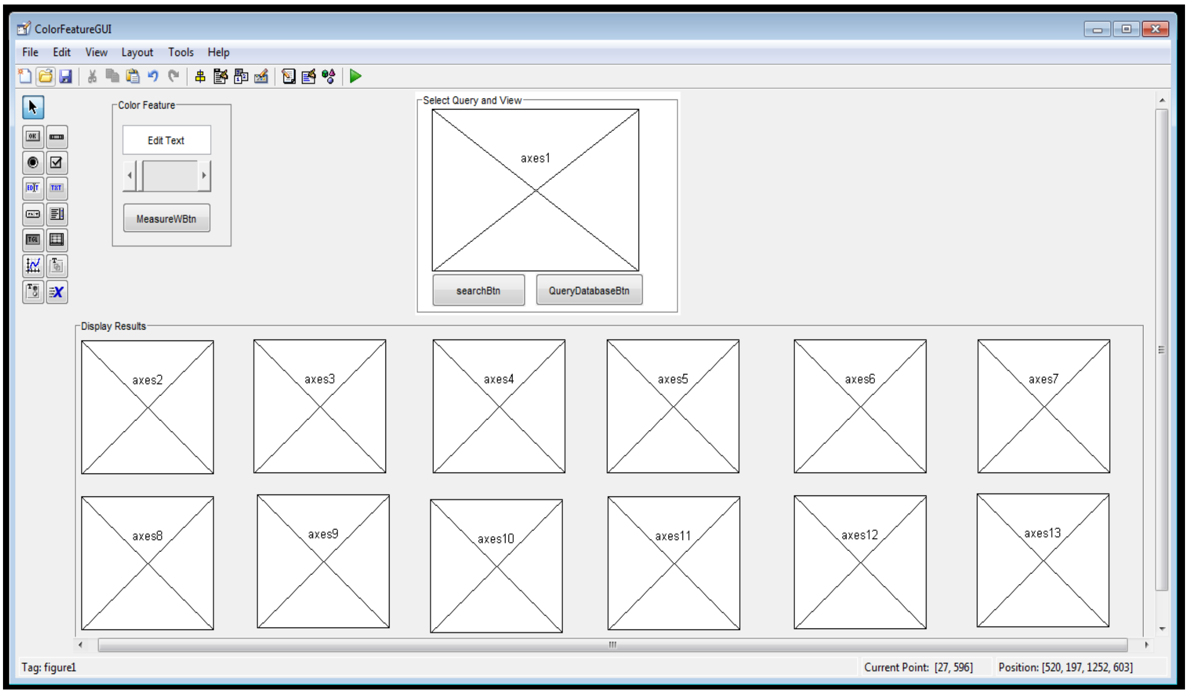


Figure 6.4: Generated GUI for Colour Feature

Colour feature extraction method

The colour feature technique implemented to compute the extracted feature vector is the colour histogram. It computes the frequency occurrence of pixels contained in each image and are stored in the database. The query image was selected from the Matlab workspace. Each image contains 255 frequency occurrences, but due to retrieval time is reduced to fewer frequencies for instance 16, 32, 64 and 128. In listing 6.1, the push button is linked to a dialog box that lists the image files in the current Matlab workspace (see Figure 6.5). The user selects a single image to allow for the computation of the image's features vectors.

```
function pushbutton1_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------------
%   when a button clicked,display a modal dialog box that lists image files
%   in the current.Matlab workspace and enable the user to select a single
%   image in order to compute image's histogram features.
% --------------------------------------------------------------------------
[FileName, folder] = uigetfile({'*.jpg'}, 'Select File');
if FileName ~= 0
fullName = fullfile(folder,FileName);
end
Masat_images = imread(fullName);
x_axis = 0:16:255;
y_axis = Masat_images(1:end);
freq = histc(y_axis,x_axis);
freq(1);freq(2); freq(3);freq(4)..........freq(16);
Masat_images = reshape(Masat_images,[],1);
% --------------------------------------------------------------------------
%   Establishes the connection to MySQL Database
% --------------------------------------------------------------------------
conn = datatabase('satelliteimageDB','root','password123');
a = isconnection(conn);
if a == 1
disp ('Mysql Database is connected.......');
end
if a == 0;
disp (' Please try again,connection not established!!!!');
end
% --------------------------------------------------------------------------
%  Creates table and store image's compute histogram features into Database
% --------------------------------------------------------------------------
exdata1 = {FileName, freq(1), freq(2), freq(3)................freq(16);
fastinsert(conn, 'Color_Feature_Vector', {'filenamedata'; 'firstbin'; 'secondbin';...
'thirdbin'; 'fourthbin'; 'fifthbin';.....'sixteenthbin'},exdata1)
exec(conn,'commit');
close (conn);
end
```

Listing 6.1: Computing Histogram Features

The values of the extracted frequencies are stored in the variables freq (1) up to freq (16). The connection to the database is established using *conn* that contains the name of the created database as well login detail and password, to allow for the computed image histogram frequency values to be stored into the database. The values are stored in the created table called Color_Feature_Vector. Its should be noted, that for research and listing purposes a default login and password is used in Listing 6.1. Proper authentication mechanisms will come into play within the final implementation.

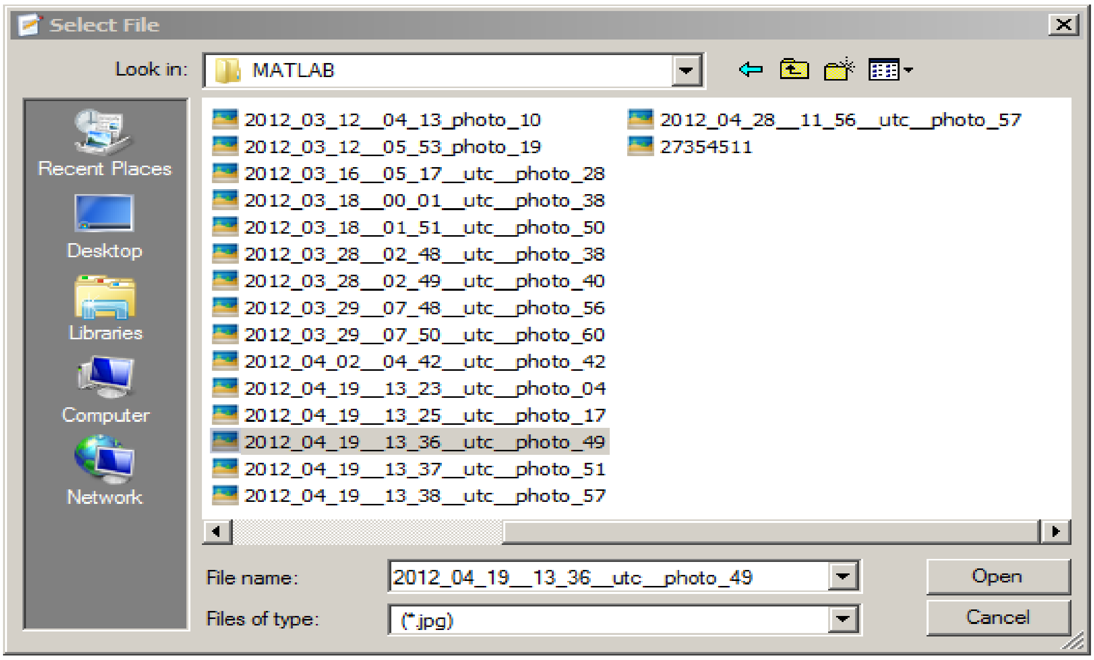


Figure 6.5: Dialog box to select single query image

Example of the extracted histogram values of an image to be stored in database is shown in Table 6.2.

Table 6.2: Occurrence frequency of image

Filename	Freq 1	Freq 2	Freq 3	Freq 4	Freq 5	Freq 6	Freq 7	Freq 8	Freq 9	Freq 10	Freq 11	Freq 12	Freq 13	Freq 14	Freq 15	Freq 16
2012_04_19_13_utc_photo_49	1	170	2672	2672	10003	15207	106469	272946	205637	96102	78971	64798	41860	14134	5876	2307
2012_03_12_04_11_photo_08	4628	3790	9593	12423	17319	29146	46902	77457	93783	97608	92605	90969	90522	89668	8096 1	4455
2012_03_12_04_13_photo_10	2939	2952	7730	16489	32350	54955	62224	75585	106425	108907	87776	74132	71064	73000	62145	3147
2012_03_29_07_50_utc_photo_60	205883	9809	22358	37873	40638	30952	29349	41882	58822	57880	60007	60139	67563	62291	70702	4172

6.3.2 Shape Feature GUI Development

Figure 6.6 shows the shape feature GUI developed using GUIDE and programming code. The GUI consists of three panels similar to those in Section 6.3.1, namely an edit text, slider and button. The purpose of edit text is to display the value of the slider ranging from 0 to 1 of the weightage of the feature. The button takes the value of the slider, combines it with the extracted value of the query image so that the search can be done against images in the database. The select query and view panel display the choice of the query image to be used as per images in the database. The panel contains two buttons: one to select the query and the second to take the query and search the database. The display results panel includes twelve axes components which are used to display the retrieved results. The associated source code of the shape feature GUI developed is listed in Appendix A: Listing A.2.

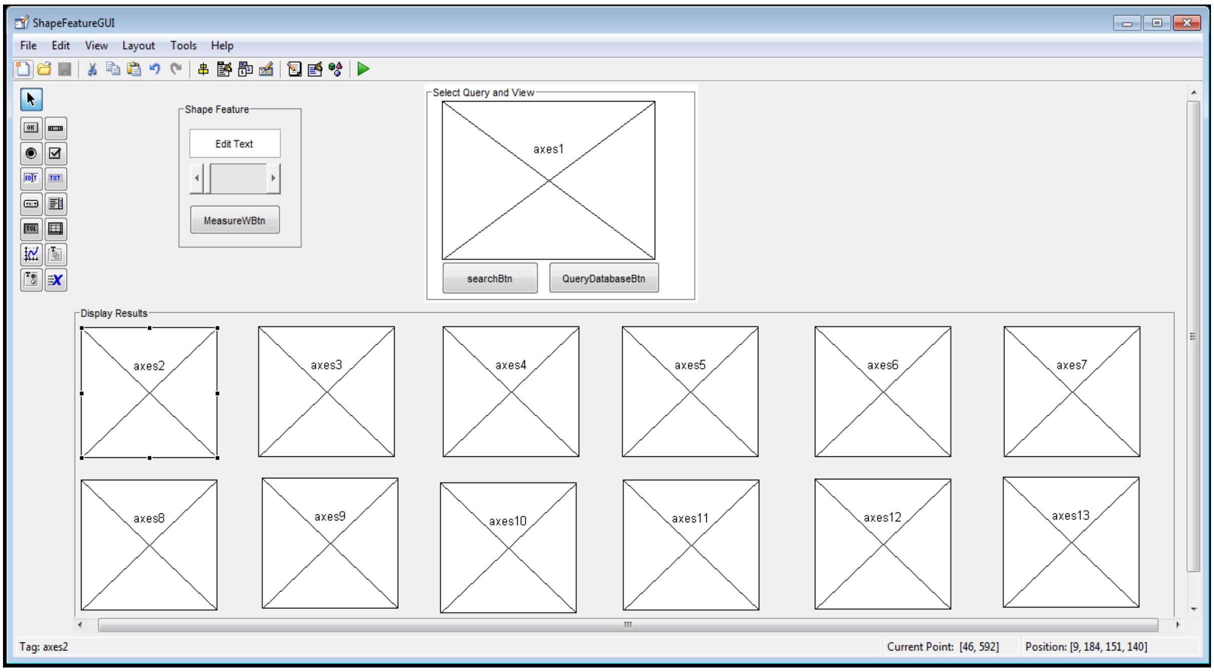


Figure 6.6: Developed GUI for shape feature

Shape feature extraction Method

The shape feature technique (Morphology operator) is used to compute all shape features of a selected image. Each time an image is chosen, the shape feature is extracted using the morphology method. Once computation is finished a connection to the database is established. Table 6.3 shows the shape feature vector stored in the database for each Masat-1 image selected.

Table 6.3: Parameters of shape feature to be store in database

Filename	Perimeter	Area	Roundness
2012_03_12__04_13_photo_10	943.1270	10661	0.1506
2012_03_12__04_11_photo_08	249.5391	1663	0.3356
2012_03_29__07_50__utc__photo_60	426.6102	3898	0.2691
2012_03_29__07_48__utc__photo_56	329.6812	2045	0.2364

In listing 6.2, an extract shape push button is linked to a dialog box that lists images in the current Matlab workspace. A choice is selected by the user and the push button is clicked in order to extract shape feature vectors to be stored, once the database connection is established.

```
function ExtractShapeBtn_Callback(hObject, eventdata, handles)
% ---------------------------------------------------------------------------
%   ExtractShapeBtn,display a modal dialog box that lists image files in the current
%   Matlab workspace and enable the user to select a single image in
%   order to compute image's shape features.
% ---------------------------------------------------------------------------
[FileName, folder] = uigetfile({'*.jpg'}, 'Select File');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end

Masat_images = imread(fullName);
converted_image = im2bw(Masat_images,0.94);
se = strel('disk',3); % create a structure element se of type of disk size 3
openedBW = imopen(converted_image,se); % performs morphological opening on the binary image
bw = bwareaopen(openedBW,1500);% removes all objects in the fewer than 1500 pixels
bw = imfill(bw,'holes'); % fills holes in the binary image bw
[B,L] = bwboundaries(bw,'noholes');% search only for object(parent and child) boundaries
stats = regionprops(L,'Area','Centroid'); % measure properies of image regions
s = 0.94;
for k = 1:length(B)
    boundary = B{k}
    delta_sq = diff(boundary).^2; % Compute a simple estimate of the object's parimeter
    perimeter = sum(sqrt(sum(delta_sq,2)));
    area = stats(k).Area; % obtain the area calculation corresponding to label k
    roundness = 4*pi*area/perimeter^2; % compute the roundness metric
    Masat_images = reshape(Masat_images,[],1);
    % ---------------------------------------------------------------------------
    %   Establishes the connection to MySQL Database
    % ---------------------------------------------------------------------------
    conn = datatabase('satelliteimageDB','root','password123');
    a = isconnection(conn);
    if a == 1
        disp ('Mysql Database is connected.......');
    end
    if a == 0;
        disp (' Please try again,connection not established!!!!');
    end
    % ---------------------------------------------------------------------------
    %  Creates table and store image's computed shape features into Database
    % ---------------------------------------------------------------------------
    exdata3 = {FileName, perimeter,area,roundness};
    fastinsert(conn, 'Shape_Feature_Vector', {'filenamedata';
'perimeter','area','roundness'},exdata3)

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

Listing 6.2: Computing shape features

6.3.3 Texture Feature GUI Development

Figure 6.7 shows the texture feature GUI developed using GUIDE and programming code. The GUI has three panels similar to those in Section 6.3.1 and 6.3.2. The texture feature panel includes an edit text, slider and button. The purpose of edit text is to display the value of slider ranging from 0 to 1 of the weightage of the feature. The value of the slider is combined it with the extracted value of the query image so that the search can be done against images in the database. The select query and view panel display the choice of the query image to be used against images in the database. The panel contains two buttons: one to select the query and the other to search the database. The display results panel includes twelve axes components that are used to display the retrieved results. The associated source code of the texture feature GUI developed can be found in Appendix A: Listing A.3.

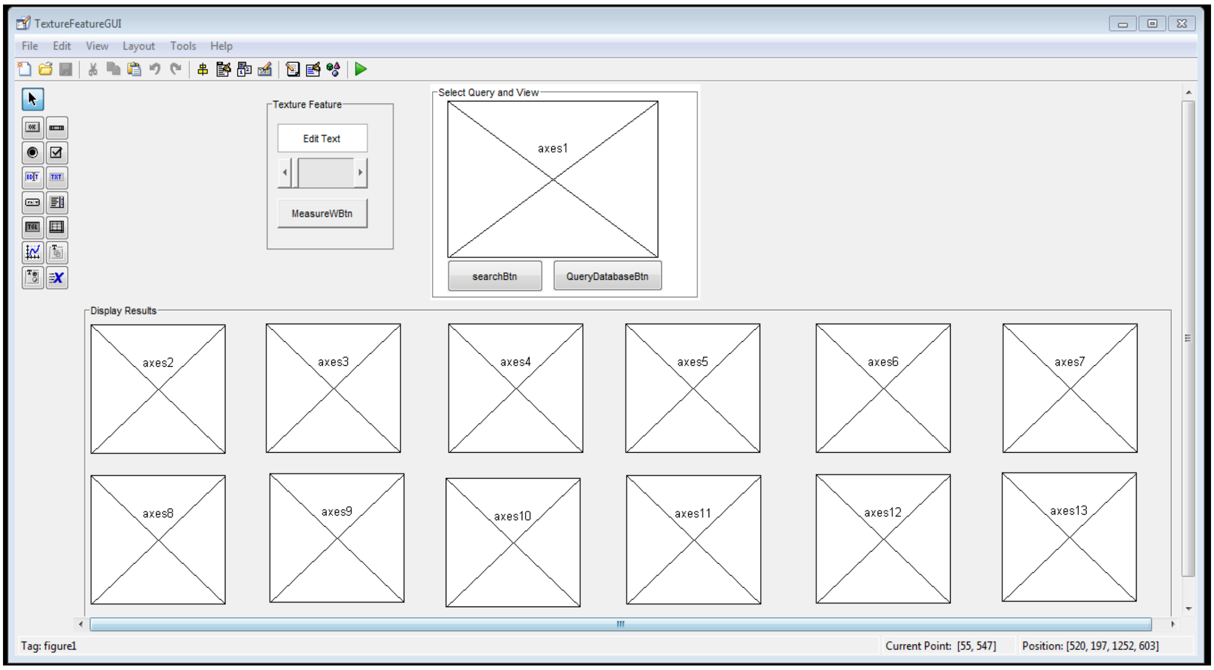


Figure 6.7: Developed GUI for Texture feature

Texture feature extraction Method

The texture feature implemented is the gray level co-occurrence matrix (GLCM). To compute GLCM, the Graycomatrix function is used to extract texture features. This function calculates intensity contained in the image and can reduce the intensity value using a default scale of 8 and of a single GLCM. However, a single GLCM is not enough to describe all features contained in an individual image. A multiple GLCM of an array of offset values together with the Graycomatrix function can be computed, and these offsets are able to determine pixel relationships of varying direction and distance. As shown in Listing 6.3, the array has offsets (i.e. 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]) that indicate four directions (horizontal, vertical and two diagonals) and four distances. The input image is now represented by 16 GLCMs. When an image is selected, the image filename and four parameter values are extracted, namely mean_contrast, mean_correlation, mean_energy and mean_homogeneity.

Table 6.4: Graycoprops normalized properties

Property	Description	Formula
'Contrast'	Returns a measure of the intensity contrast between a pixel and its neighbor over the whole image. `Range = [0 (size(GLCM,1)-1)^2]` Contrast is 0 for a constant image.	$\sum_{i,j} \lvert i-j \rvert^2 p(i,j)$
'Correlation'	Returns a measure of how correlated a pixel is to its neighbor over the whole image. `Range = [-1 1]` Correlation is 1 or -1 for a perfectly positively or negatively correlated image. Correlation is `NaN` for a constant image.	$\sum_{i,j} \frac{(i-\mu i)(j-\mu j)p(i,j)}{\sigma_i \sigma_j}$
'Energy'	Returns the sum of squared elements in the GLCM. `Range = [0 1]` Energy is 1 for a constant image.	$\sum_{i,j} p(i,j)^2$
'Homogeneity'	Returns a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. `Range = [0 1]` Homogeneity is 1 for a diagonal GLCM.	$\sum_{i,j} \frac{p(i,j)}{1+\lvert i-j \rvert}$

Table 6.4, shows the graycoprops calculation specified in properties such as contrast, correlation, energy and homogeneity. Graycoprops normalizes the GLCM properties to a value of 1.

```
function ExtractTextureBtn_Callback(hObject, eventdata, handles)
% -------------------------------------------------------------------------
%  ExtractTextureBtn,display a modal dialog box that lists image files in the current
%   Matlab workspace and enable the user to select a single image in
%   order to compute image's Texture features.
% -------------------------------------------------------------------------
[FileName, folder] = uigetfile({'*.jpg'}, 'Select File');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
Masat_images = imread(fullName);

GLCMS = graycomatrix(Masat_images,'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');
sum_contrast = 0;
for i = 1:16
    sum_contrast = sum_contrast + contrast.Contrast(i);
    mean_contrast = sum_contrast/16;
end
mean_contrast;
sum_correlation = 0;
for i = 1:16
    sum_correlation = sum_correlation + correlation.Correlation(i);
    mean_correlation = sum_correlation/16;
end
mean_correlation;
sum_homogeneity = 0;
for i = 1:16
    sum_homogeneity = sum_homogeneity + homogeneity.Homogeneity(i);
    mean_homogeneity = sum_homogeneity/16;
end
mean_homogeneity;
sum_energy = 0;
for i = 1:16
    sum_energy = sum_energy + energy.Energy(i);
    mean_energy = sum_energy/16;
end
mean_energy;
     Masat_images = reshape(Masat_images,[],1);
% -------------------------------------------------------------------------
%   Establishes the connection to MySQL Database
% -------------------------------------------------------------------------
conn = datatabase('satelliteimageDB','root','password123');
a = isconnection(conn);
if a == 1
    disp ('Mysql Database is connected.......');
end
if a == 0;
    disp (' Please try again,connection not established!!!!');
end
% -------------------------------------------------------------------------
%  Creates table and store image's computed texture features into Database
% -------------------------------------------------------------------------
exdata2 = {FileName, mean_contrast,mean_correlation,mean_homogeneity,mean_energy};
fastinsert(conn, 'Texture_Feature_Vector', {'filenamedata';
'contrast','correlation','homogeneity','energy'},exdata2)

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

Listing 6.3: Computing Texture features

6.3.4 Data Classify Panel GUI Development

The obtained images are embedded with information, such as GPS co-ordinates, and includes the time and date of when the image was taken.The function *imfinfo* is used to obtain Exchangeable image file format (EXIF) metadata such as filename, date and time (see Figure 6.8).

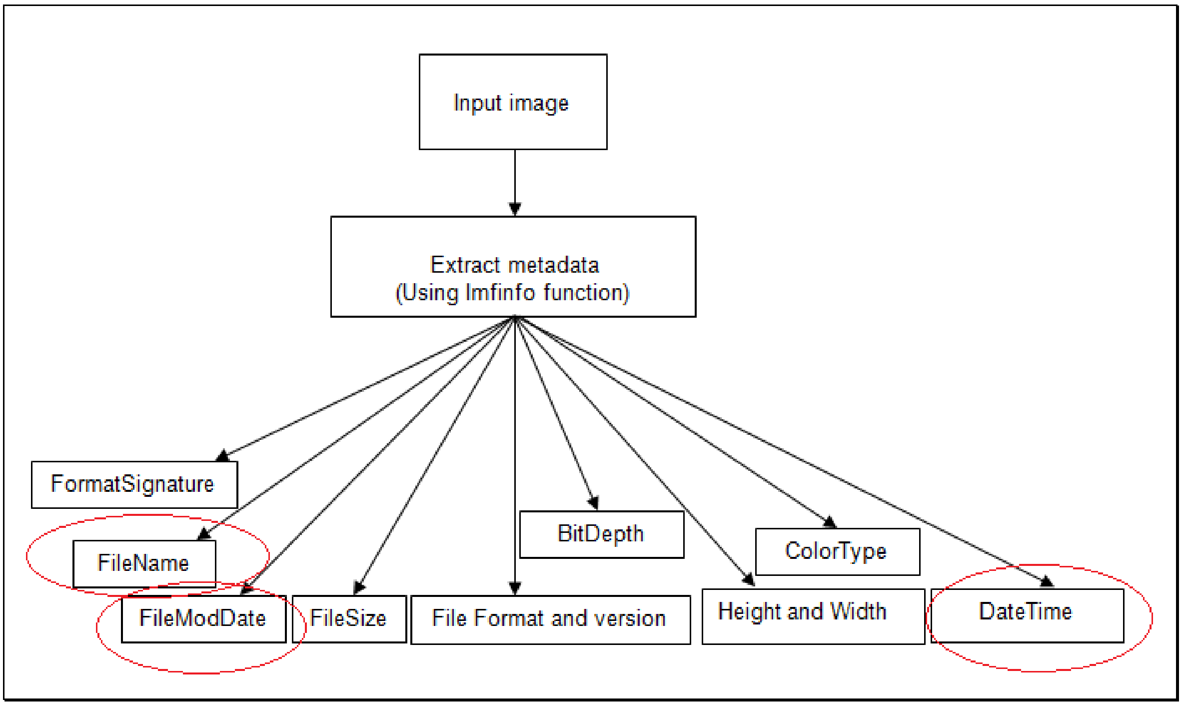


Figure 6.8: Extracting image metadata

In listing 6.4, the source code for extracting the metadata is shown. Users select the images through a standard dialog box containing all images which are loaded in the MATLAB workspace. The dialog box is generated by the uigetfile function and a default JPEG format is used (other formats such as GIF, TIFF can also be used depending on the applications). Two parameters, FileName and Folder, are received from uigetfile function. The fullfile stores the equivalent input argument strings of the image filename and folder. The the first loop uses the imfinfo function to extract an image and store it in a variable named value1. A structured array is then created with the variable named Ti to store the content of value1 of the extracted image content. The second loop returns the content of the filename and FileModDate fields using Fn and Dn respectively. The tf variable structure examines whether a filename exists of the selected image. A logical value of 1 is given if it does exist and 0 if doesn't exist. When the filename exists, both the filename and FileModDate are stored in the structure called main. Finally, a filename of the select image, date and time are extracted. For the complete source code of the project, including storing extracted metadata into the MySQL database, refer to Appendix A: Listing A.4. The database is designed to store images according to the months they were captured.

```
function ExtractMetadata_Callback(hObject, eventdata, handles)
% -----------------------------------------------------------------------
%   Extraction of image metadata using Filename and FileModDate. Initially,
%   Dialog box will be displayed to select image to extract metadata.
%------------------------------------------------------------------------

[FileName, folder] = uigetfile({'*.jpg'}, 'Select File');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
img = imread(fullName);
img = reshape(img,[],1);
A = FileName;
B = folder;
[m,n]=size(A);
for i=1:n
    Value1 = imfinfo(A);
    S(i) = struct('Ti',Value1);
    S(i);
end
for i=1:n
    Fn{i}=getfield(S(1,i).Ti,{1,1},'Filename');
    Dn{i}=getfield(S(1,i).Ti,{1,1},'FileModDate');
    tf = isfield(S(1,i).Ti,'Filename');
    if tf==1
        Fn{i}=getfield(S(1,i).Ti,{1,1},'Filename');
    else Fn{i}=char('data not available');
    end
    main = struct('File',Fn{i},'DatabaseOfImages',Dn{i});
    var = struct2cell(main);
end
Dtime = datenum(Value1.FileModDate);
datestr(Dtime);
[y, m, d, h, mn, s] = datevec(Dtime);
sprintf('Date: %d/%d%d',y,m,d);
```

Listing 6.4: Extracting of filename and FileModDate of the image

6.3.5 IntegratedGUI Development

Figure 6.9 shows a GUI was developed to combine all the feature extraction methods, such as colour, shape, texture and date classifier. The GUI has edit texts for all features to allow the user to view the selected values to sum all values together with the selected image so that similar images can be retrieved. The associated source code of the integrated GUI is available in Appendix A: Listing A.5.

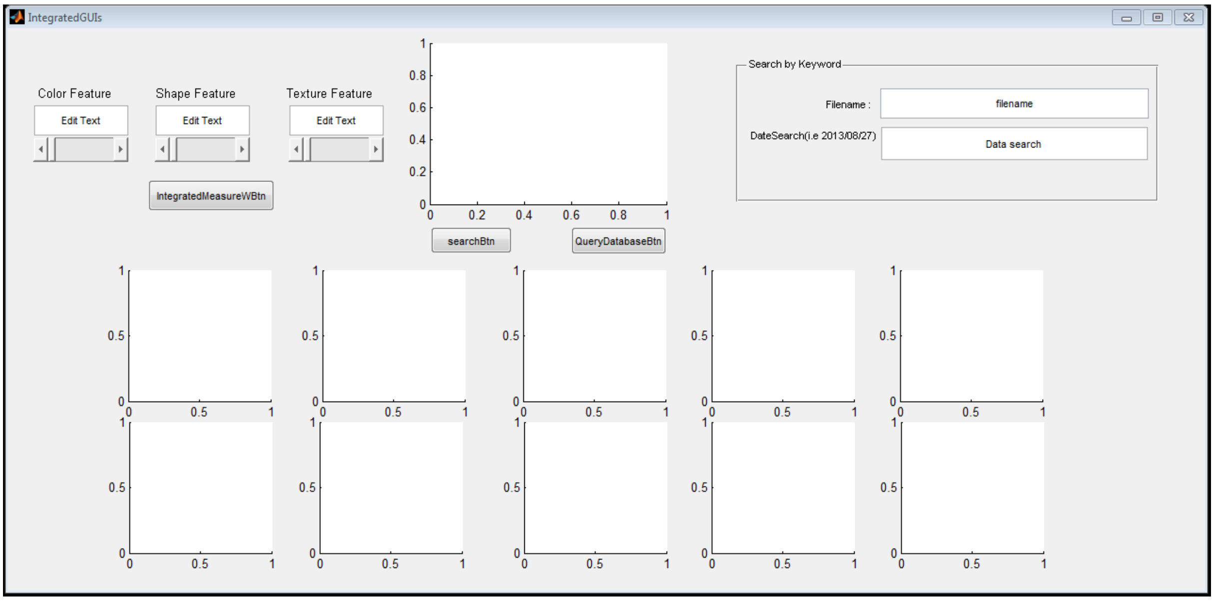


Figure 6.9: IntegratedGUIs

6.4 Database development and Designing

In this section, the database is developed as shown in Figure 6.10. Initially, the features of each image are extracted and stored into appropriate tables. In this project, forty tables were created (i.e., Color_Features_Vector, Shape_Feature_Vector and Texture_Feature_Vector).

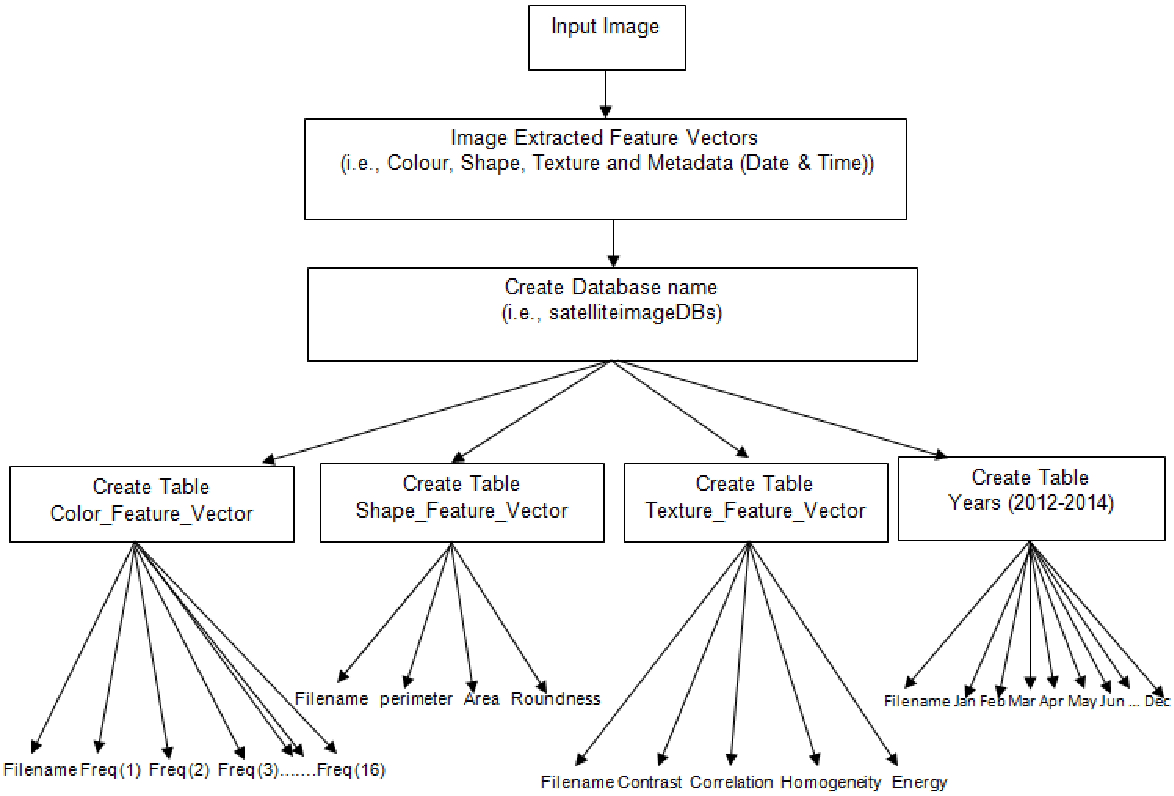


Figure 6.10: Block diagram of creating database and tables of the project

6.4.1 Designing a Database

In this project, only one database was created to hold the tables to store all original images, metadata and their feature vectors. MySQL relational database was created called *satelliteimageDBs* using standard SQL programming language (see Figure 6.11 creating database).

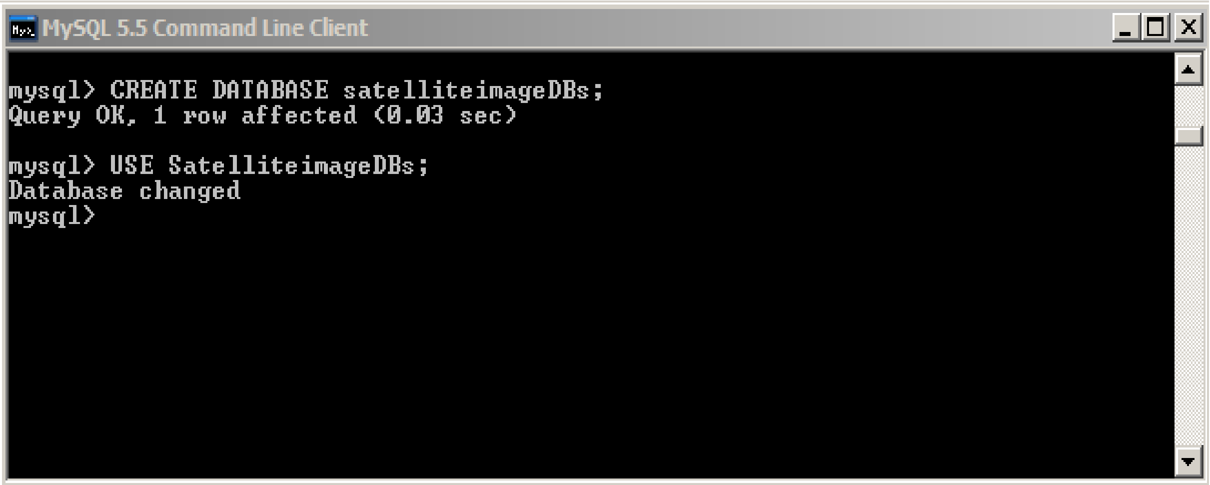


Figure 6.11: Creating Database for the project

The type of table used in this project is the default table InnoDB of the MySQL 5.5 database or latest version. It is a table that supports a low level locking, foreign key constraints and multiversioning. Figure 6.12 details the Color_Feature_Vector table that stores all the colour histogram values.

```
MySQL 5.5 Command Line Client
mysql>
mysql> CREATE TABLE Color_Feature_Vector (
    -> filename VARCHAR(255) NOT NULL,
    -> freq1 INT NOT NULL,
    -> freq2 INT NOT NULL,
    -> freq3 INT NOT NULL,
    -> freq4 INT NOT NULL,
    -> freq5 INT NOT NULL,
    -> freq6 INT NOT NULL,
    -> freq7 INT NOT NULL,
    -> freq8 INT NOT NULL,
    -> freq9 INT NOT NULL,
    -> freq10 INT NOT NULL,
    -> freq11 INT NOT NULL,
    -> freq12 INT NOT NULL,
    -> freq13 INT NOT NULL,
    -> freq14 INT NOT NULL,
    -> freq15 INT NOT NULL,
    -> freq16 INT NOT NULL,
    -> PRIMARY KEY(filename)
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> describe Color_Feature_Vector;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| filename | varchar(255) | NO   | PRI | NULL    |       |
| freq1    | int(11)      | NO   |     | NULL    |       |
| freq2    | int(11)      | NO   |     | NULL    |       |
| freq3    | int(11)      | NO   |     | NULL    |       |
| freq4    | int(11)      | NO   |     | NULL    |       |
| freq5    | int(11)      | NO   |     | NULL    |       |
| freq6    | int(11)      | NO   |     | NULL    |       |
| freq7    | int(11)      | NO   |     | NULL    |       |
| freq8    | int(11)      | NO   |     | NULL    |       |
| freq9    | int(11)      | NO   |     | NULL    |       |
| freq10   | int(11)      | NO   |     | NULL    |       |
| freq11   | int(11)      | NO   |     | NULL    |       |
| freq12   | int(11)      | NO   |     | NULL    |       |
| freq13   | int(11)      | NO   |     | NULL    |       |
| freq14   | int(11)      | NO   |     | NULL    |       |
| freq15   | int(11)      | NO   |     | NULL    |       |
| freq16   | int(11)      | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
17 rows in set (0.00 sec)
```

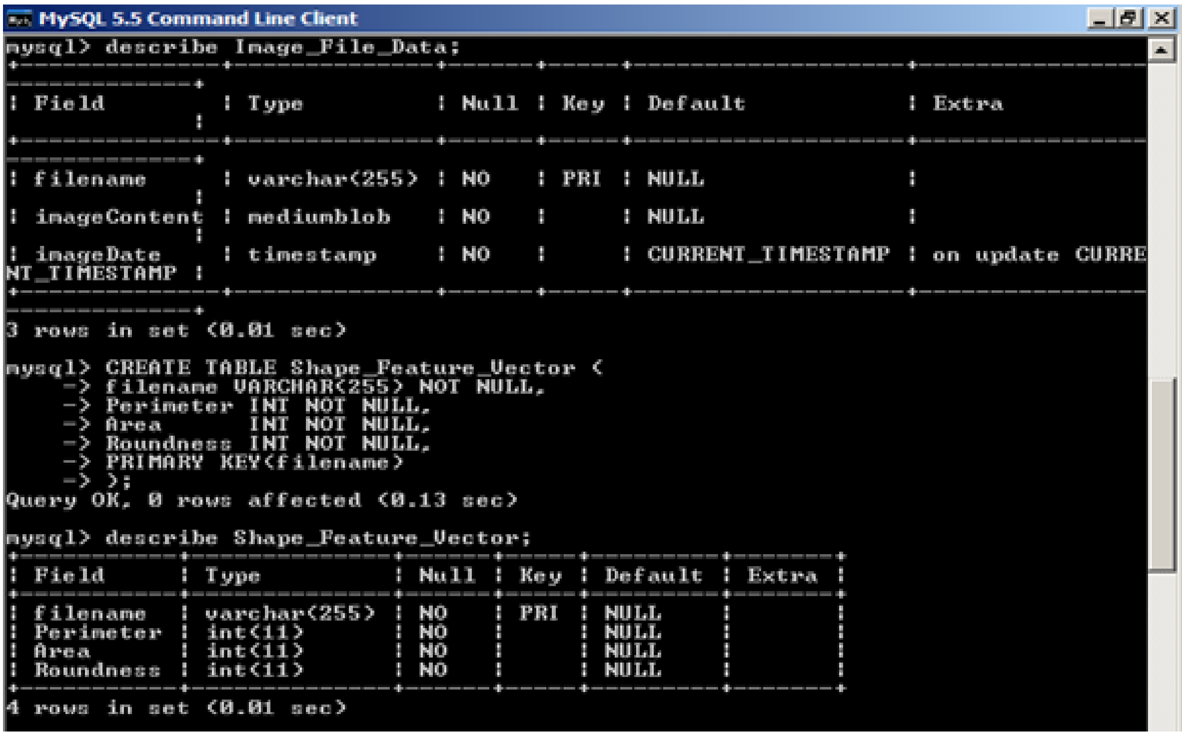
Figure 6.12: Color feature vector table

The satellite images captured are stored in the table called image file data (see Figure 6.13) without extracting any content.

```
MySQL 5.5 Command Line Client
mysql> \c
mysql>
mysql>
mysql>
mysql> CREATE TABLE Image_File_Data (
    -> filename VARCHAR(255) NOT NULL,
    -> imageContent MEDIUMBLOB NOT NULL,
    -> imageDate TIMESTAMP NOT NULL,
    -> PRIMARY KEY(filename)
    -> );
Query OK, 0 rows affected (0.65 sec)

mysql> describe Image_File_Data;
+--------------+--------------+------+-----+-------------------+-----------------
-------------+
| Field        | Type         | Null | Key | Default           | Extra
             |
+--------------+--------------+------+-----+-------------------+-----------------
-------------+
| filename     | varchar(255) | NO   | PRI | NULL              |
             |
| imageContent | mediumblob   | NO   |     | NULL              |
             |
| imageDate    | timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRE
NT_TIMESTAMP |
+--------------+--------------+------+-----+-------------------+-----------------
-------------+
3 rows in set (0.01 sec)
```

Figure 6.13: Created image file data table to store original images

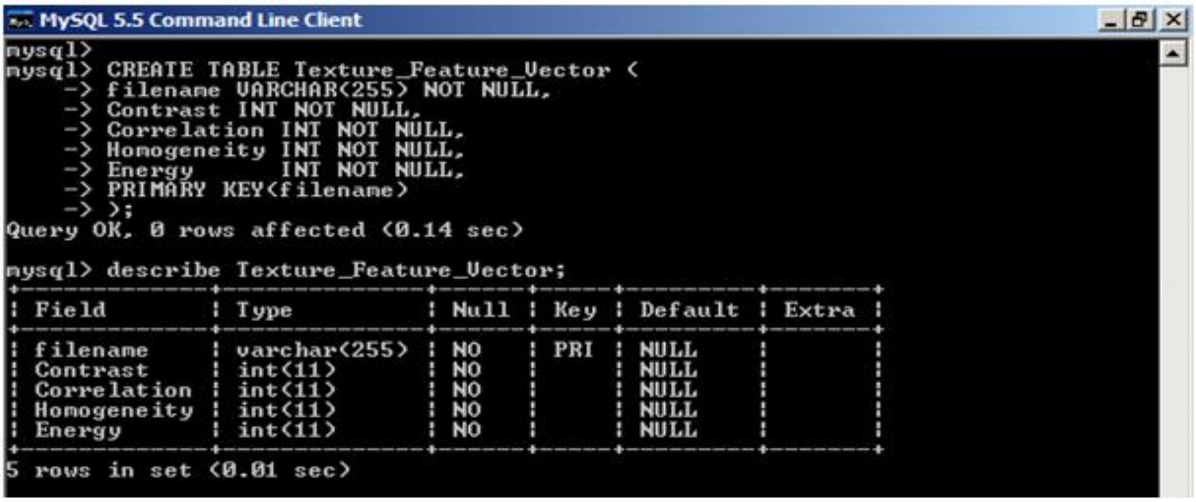
The shape feature is called Shape_Feature_Vector table stores the shape features values (see Figure 6.14).



```
mysql> describe Image_File_Data;
+--------------+--------------+------+-----+-------------------+-----------------------------+
| Field        | Type         | Null | Key | Default           | Extra                       |
+--------------+--------------+------+-----+-------------------+-----------------------------+
| filename     | varchar(255) | NO   | PRI | NULL              |                             |
| imageContent | mediumblob   | NO   |     | NULL              |                             |
| imageDate    | timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+--------------+--------------+------+-----+-------------------+-----------------------------+
3 rows in set (0.01 sec)

mysql> CREATE TABLE Shape_Feature_Vector (
    -> filename VARCHAR(255) NOT NULL,
    -> Perimeter INT NOT NULL,
    -> Area      INT NOT NULL,
    -> Roundness INT NOT NULL,
    -> PRIMARY KEY(filename)
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> describe Shape_Feature_Vector;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| filename  | varchar(255) | NO   | PRI | NULL    |       |
| Perimeter | int(11)      | NO   |     | NULL    |       |
| Area      | int(11)      | NO   |     | NULL    |       |
| Roundness | int(11)      | NO   |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Figure 6.14: Created Shape_Feature_Vector to store all images shape features

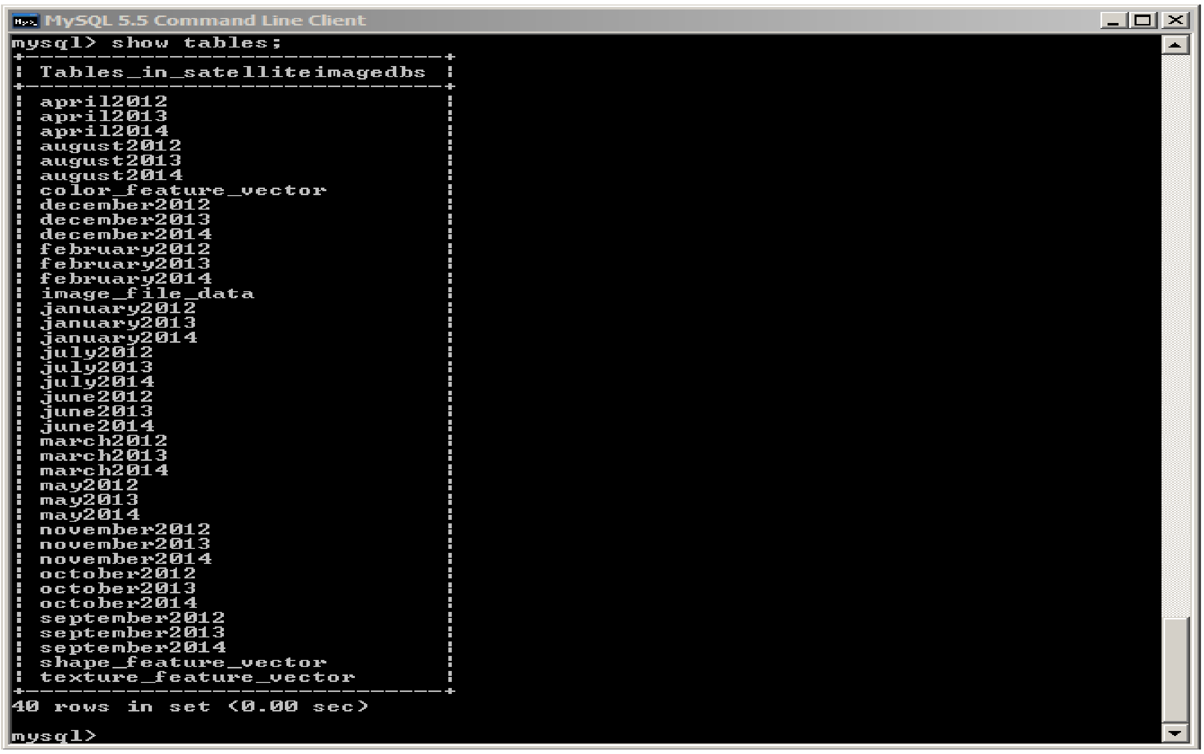
In order to store all image texture features, all column names and data types must be chosen appropriately. In case of storing filename, a character string with variable length is chosen. In Figure 6.15, a texture feature table is created. The table contains five columns in which all image names and their corresponding feature vector values are to be stored. The column names are contrast, correlation, homogeneity and energy of the computed or extracted gray level co-occurrence matrix (GLCM).



```
mysql>
mysql> CREATE TABLE Texture_Feature_Vector (
    -> filename VARCHAR(255) NOT NULL,
    -> Contrast INT NOT NULL,
    -> Correlation INT NOT NULL,
    -> Homogeneity INT NOT NULL,
    -> Energy      INT NOT NULL,
    -> PRIMARY KEY(filename)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> describe Texture_Feature_Vector;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| filename    | varchar(255) | NO   | PRI | NULL    |       |
| Contrast    | int(11)      | NO   |     | NULL    |       |
| Correlation | int(11)      | NO   |     | NULL    |       |
| Homogeneity | int(11)      | NO   |     | NULL    |       |
| Energy      | int(11)      | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Figure 6.15: Texture feature vector table

Figure 6.16 lists the satellite database containing all tables created for this project.



```
MySQL 5.5 Command Line Client
mysql> show tables;
+-----------------------------+
| Tables_in_satelliteimagedbs |
+-----------------------------+
| april2012                   |
| april2013                   |
| april2014                   |
| august2012                  |
| august2013                  |
| august2014                  |
| color_feature_vector        |
| december2012                |
| december2013                |
| december2014                |
| february2012                |
| february2013                |
| february2014                |
| image_file_data             |
| january2012                 |
| january2013                 |
| january2014                 |
| july2012                    |
| july2013                    |
| july2014                    |
| june2012                    |
| june2013                    |
| june2014                    |
| march2012                   |
| march2013                   |
| march2014                   |
| may2012                     |
| may2013                     |
| may2014                     |
| november2012                |
| november2013                |
| november2014                |
| october2012                 |
| october2013                 |
| october2014                 |
| september2012               |
| september2013               |
| september2014               |
| shape_feature_vector        |
| texture_feature_vector      |
+-----------------------------+
40 rows in set (0.00 sec)

mysql>
```

Figure 6.16: Forty tables created in the database for storing all images and metadata

6.5 Testing and Experimental Results

To ensure that the algorithms developed for this project are accurate, testing tools are developed to verify and test the accuracy of extracting and developing GUIs for each of the four algorithms.

6.5.1 Testing

To test the proposed design in section 5.4, each component and corresponding interface that was implemented needs to be tested for their functionality. Software testing is a process of running a software program for the purpose of finding errors. There are two categories of testing: verification and validation. Verification checks whether the system built is correct according to specifications, and validation refers to whether the correct specifications were followed to build the product.

A number of things needs to be considered in this project when testing an input image, such as format, size and availability of database. The software testing conducted is based on image classification, feature extraction, as well as storing and retrieving images using a query sample. Initially, images are to be received from the satellite. Once images are received, image noise is removed and the quality of the images is improved. Images are then allocated into the database. As per each algorithm, tables are populated with feature vector values extracted from the images. A number of software tools were developed in order to verify and validate functional and non-functional requirements (refer to Section 5.3) and the proposed design components.

In Figure 6.17 (a), (b) and (c), a testing tool was developed to improve the quality of images being selected as a query sample. Various image pre-processing techniques such as Median filter, Wiener Filter and histogram equalization were implemented and compared in terms of eliminating image noise. The Wiener filter outperformed both median and histogram as it improved the quality of the image better than the other methods (see Source code Appendix A: Listing A.7).

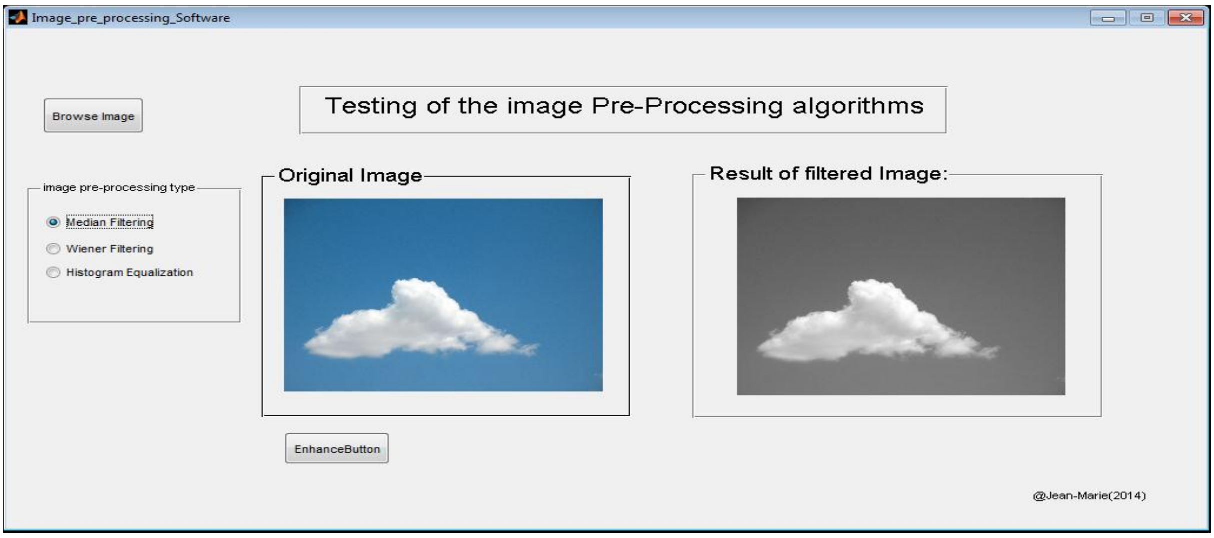


Figure 6.17(a): Testing pre-processing median filter algorithm

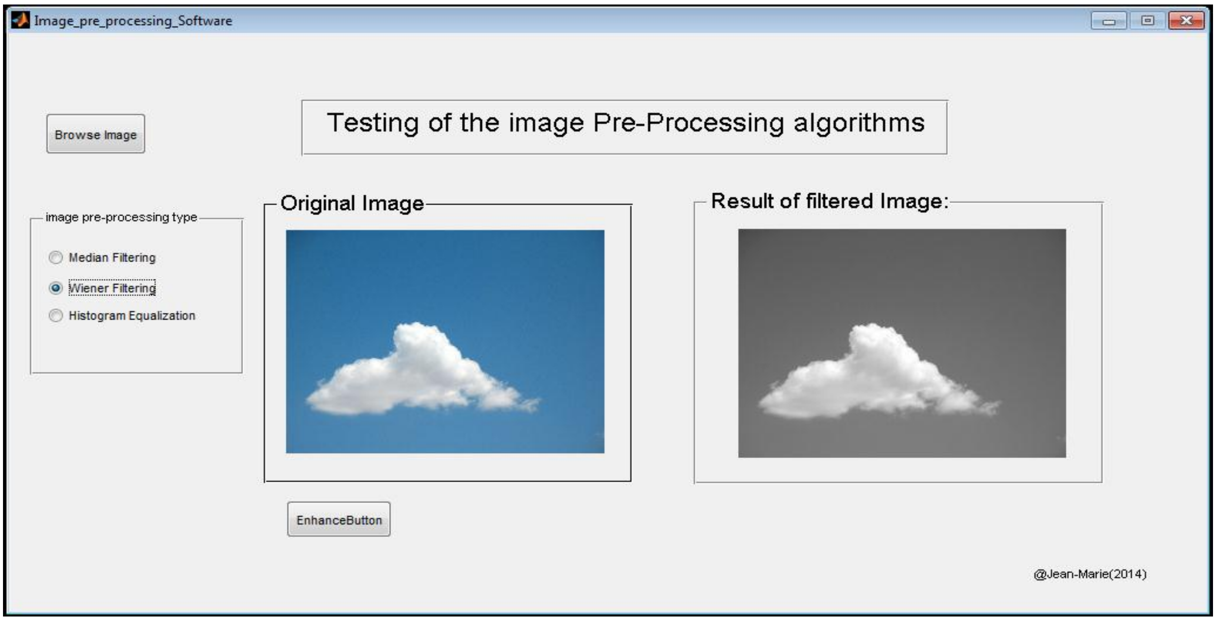


Figure 6.17(b): Testing pre-processing Wiener filter algorithm

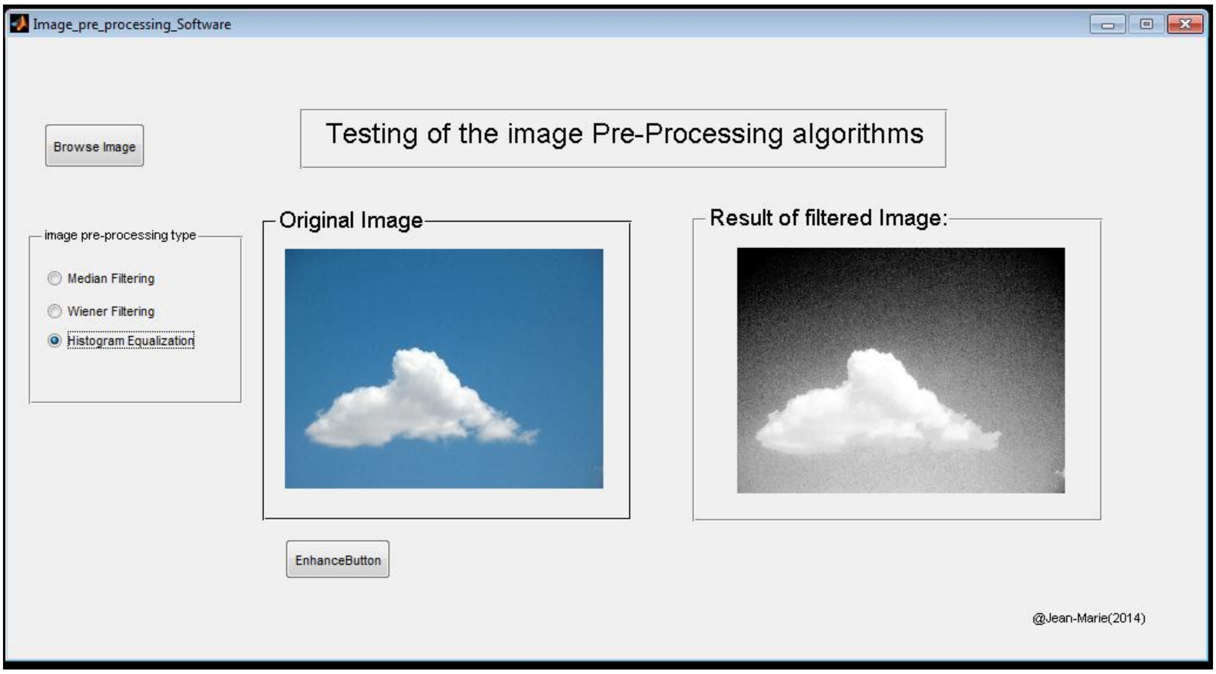


Figure 6.17(c): Testing pre-processing Histogram Equalization algorithm

In Figure 6.18, an image histogram extractor tool was developed to allow the user to input an image and to view the selected image's histogram. Usually, an image consists of 255 pixels and each bar in the histogram represents pixel colour content. Thus, each image has a histogram of 255 bars. Storing all 255 bars will slow retrieval time. Therefore, 255 bars needed to be reduced to ensure that retrieval time is improved (see source code Appendix A: Listing A.8).

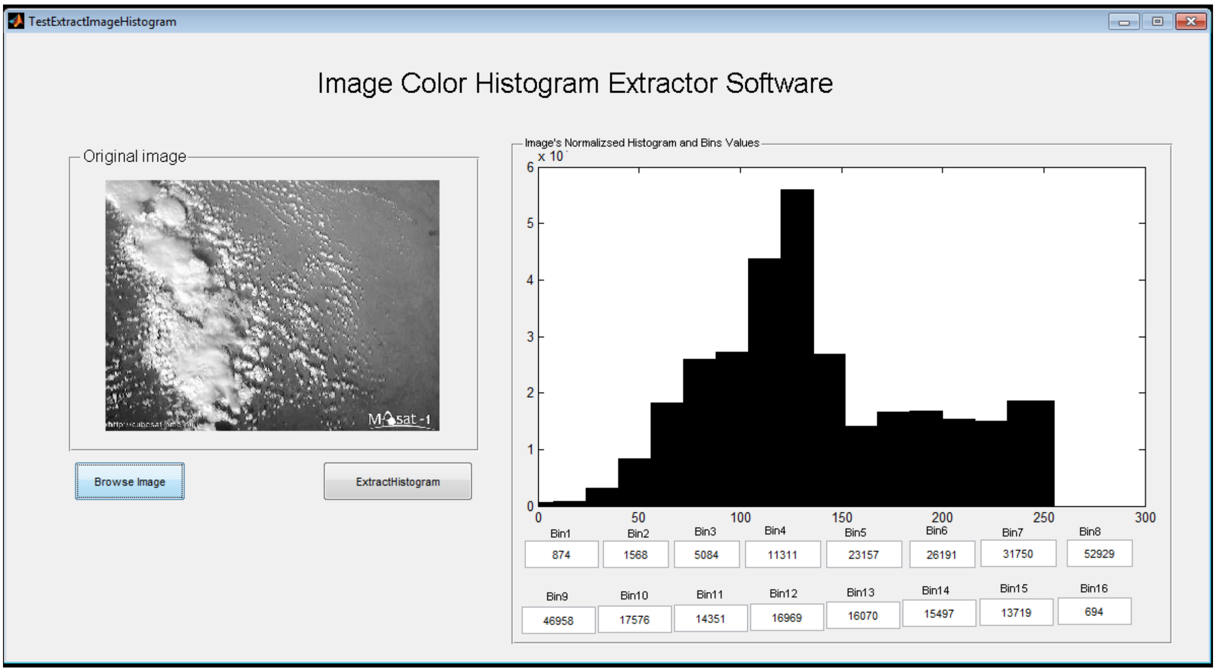


Figure 6.18: Testing colour algorithm

In Figure 6.19 a software tool was developed to test texture feature extraction using the GLCM technique. The user interface has two buttons: a browse button and extractglcmfeature button. The function of the browse button is to browse a list of images, where a user decides which image is to be used. The selected image is then displayed in a panel for viewing purposes. The second button allows the user to extract four parameter values such as contrast, correlation, energy and homogeneity. Note, that the four texture parameter values extracted are stored in created table called Texture_Feature_Vector and this table is used to compare the texture of the query (image) against of those stored texture features (see source code Appendix A: Listing A.11).

.

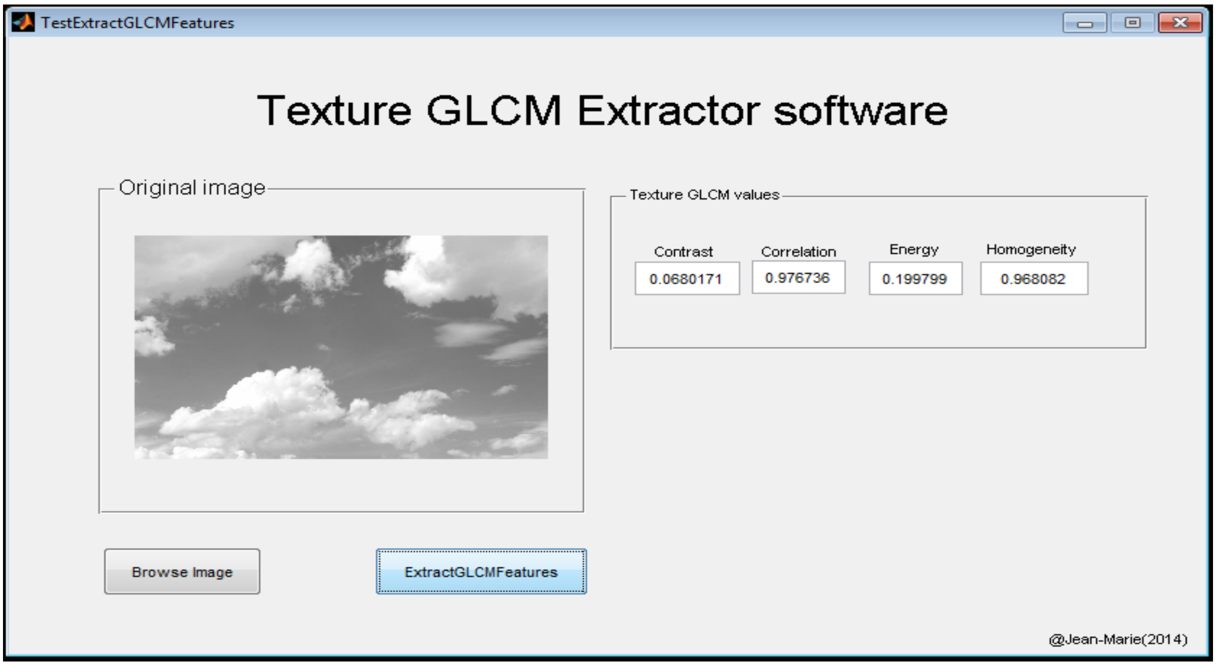


Figure 6.19: Texture Feature extraction algorithm

In Figure 6.20, a shape software tool was developed to extract three parameter values, namely perimeter, area and roundness of objects in an image. A browse button is used to open and display an image in the original image panel, and an extract morphology features button assists with the extraction of the three image features (see source code Appendix A: Listing A.10).

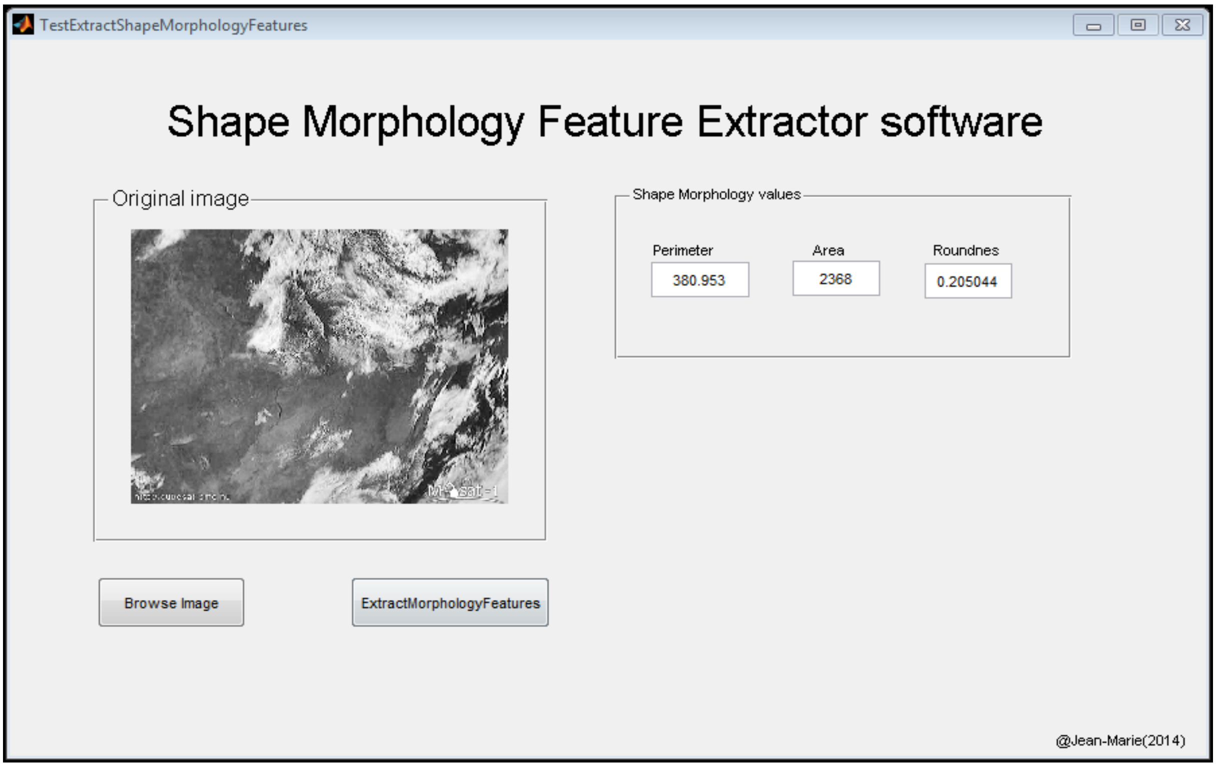


Figure 6.20: Shape Feature extraction algorithm

In Figure 6.21 (a) and (b), an Image metadata tool was developed to allow the user to select an image and extract the date and time of each image selected. Once image metadata is extracted, the image filename corresponding to the selected image is displayed. The date and time is extracted and displayed as shown. According to the images selected (see Figure 6.21 (a) and (b) extracted in figures) the date and time shows that both images were taken the same day at 11:34:32 and 11:36:14 respectively (see source code Appendix A: Listing A.9).

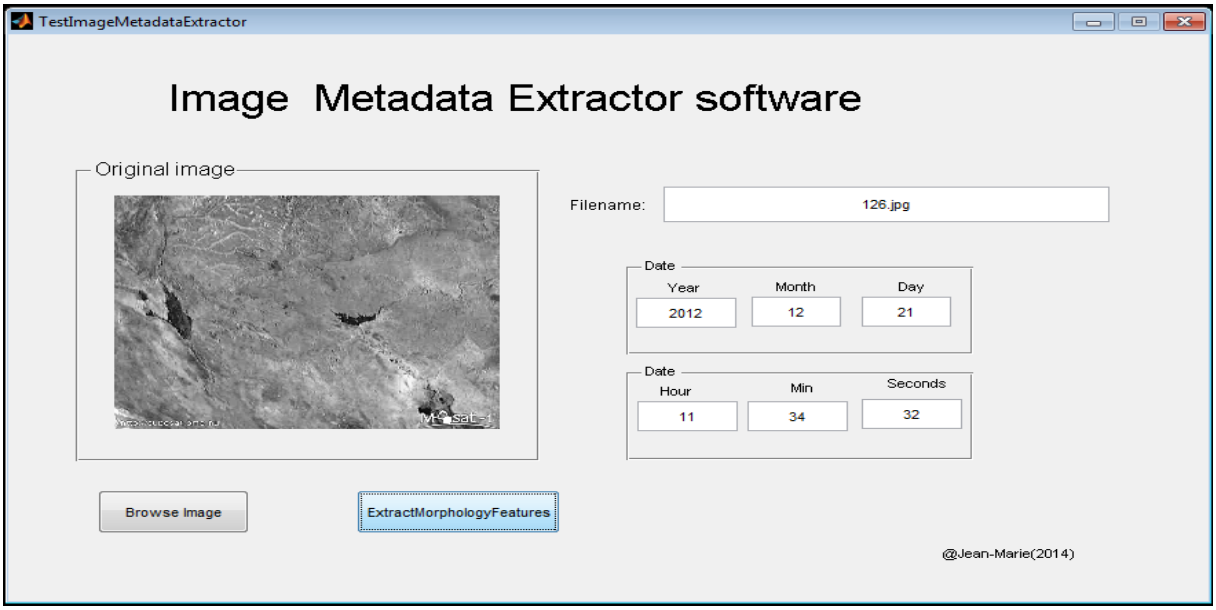


Figure 6.21(a): The image Metadata extractor algorithm

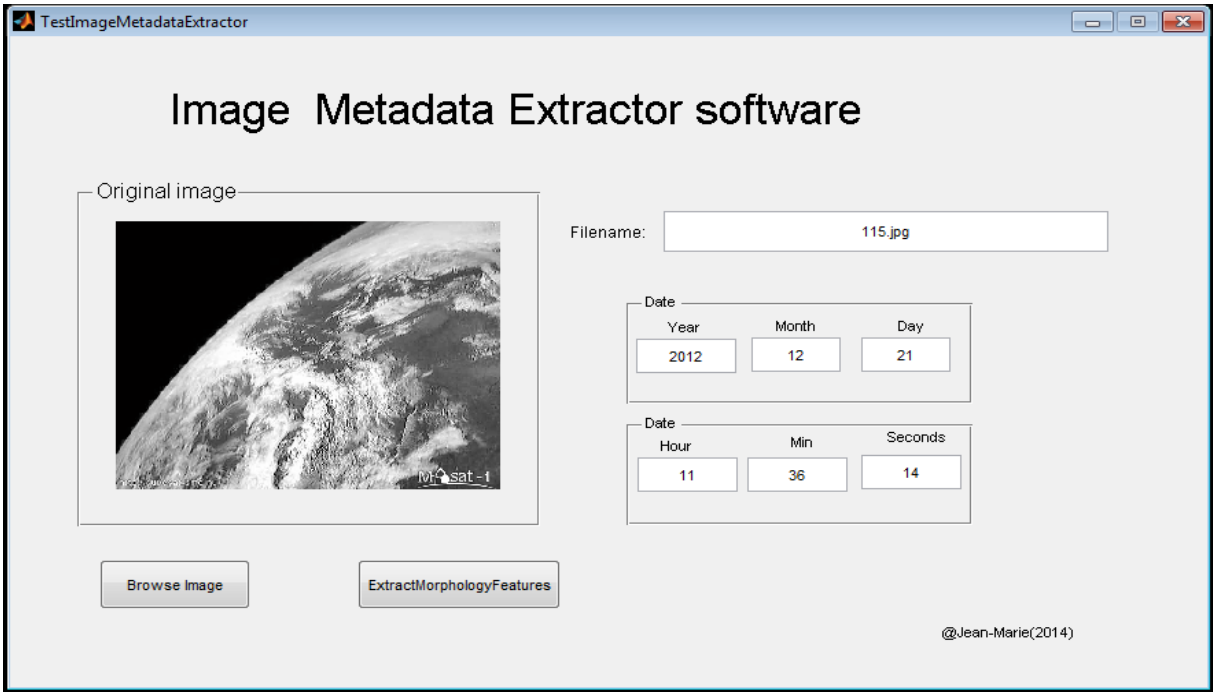


Figure 6.21(b): The image Metadata extractor algorithm

6.5.3 Experimental Results

The input image (Masat-1 Satellite images of high resolution) was used to test the functionality of each component by extracting all appropriate features. Figure 6.22 shows the results of the query developed colour feature user interface. It relies on the colour algorithm features extractor tool discussed in section 6.5.1. The histogram of each image parameter was extracted and stored into database. A query image feature was compared to stored images' features in the database. In the figure below, displayed results of the query against stored images are shown.

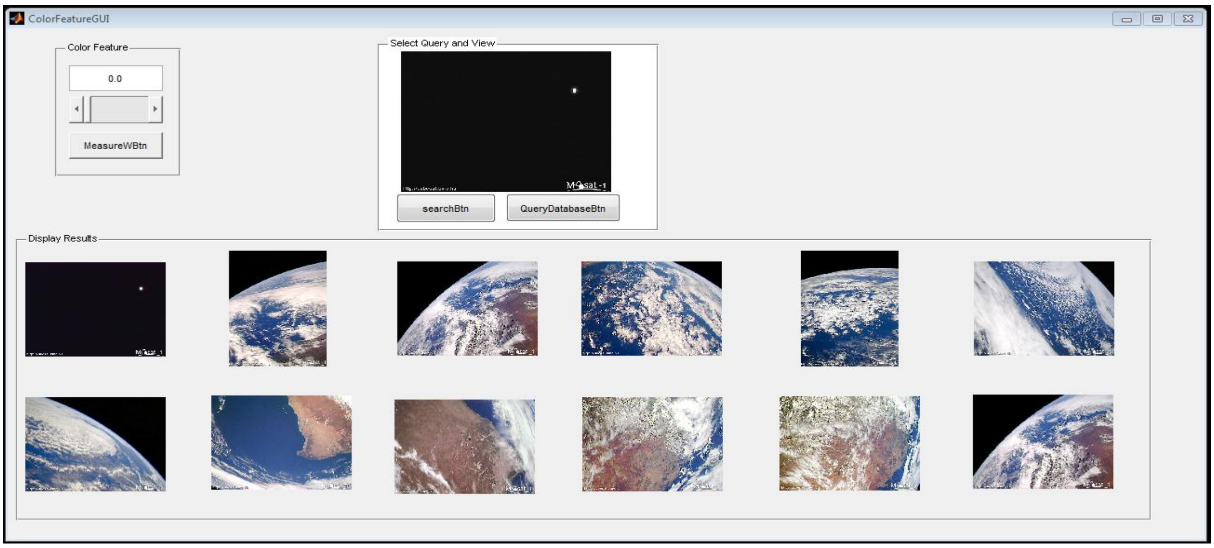


Figure 6.22: Colour Feature Results

Figure 6.23 shows the experimental results of the shape feature algorithm. The images are queried according to extracted feature parameters.

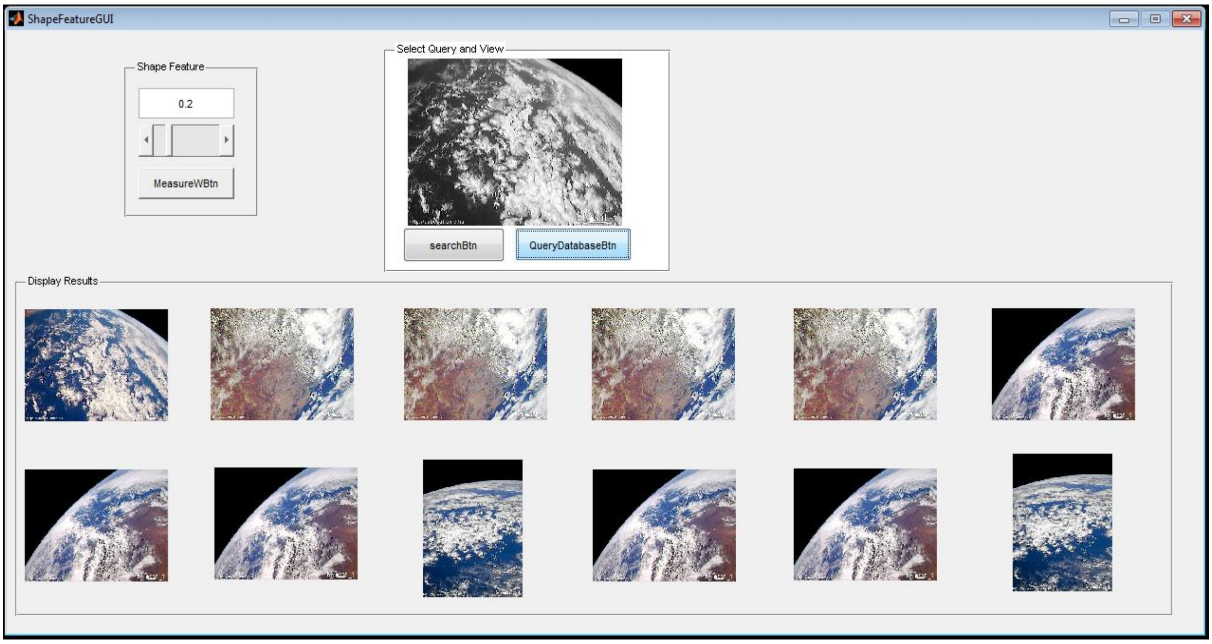


Figure 6.23: Shape Feature Results

Experimental results of the texture feature implementation is highlighted in Figure 6.24. The developed texture extractor tool implemented the texture feature algorithm and displays the queried results.

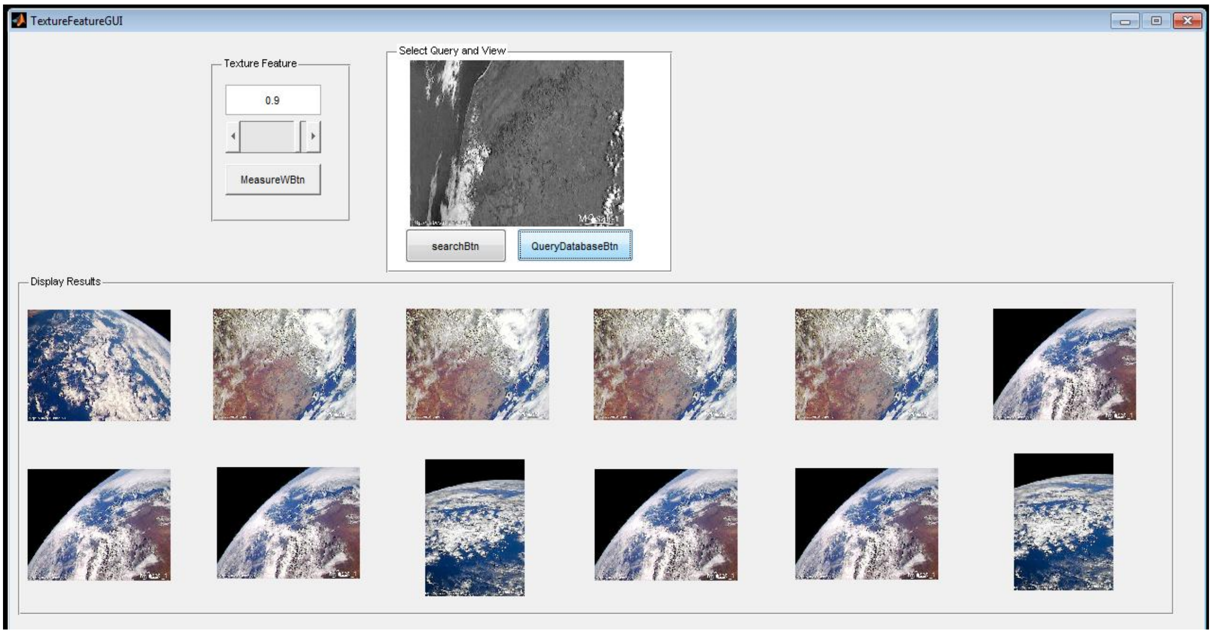


Figure 6.24: Texture Feature Result

An integrated graphical user interface was developed whereby all Colour, Shape, Texture and image metadata features were integrated into one user interface (Figure 6.25). In the figure a panel was provided for the user to select the desired weight age value for all three features.

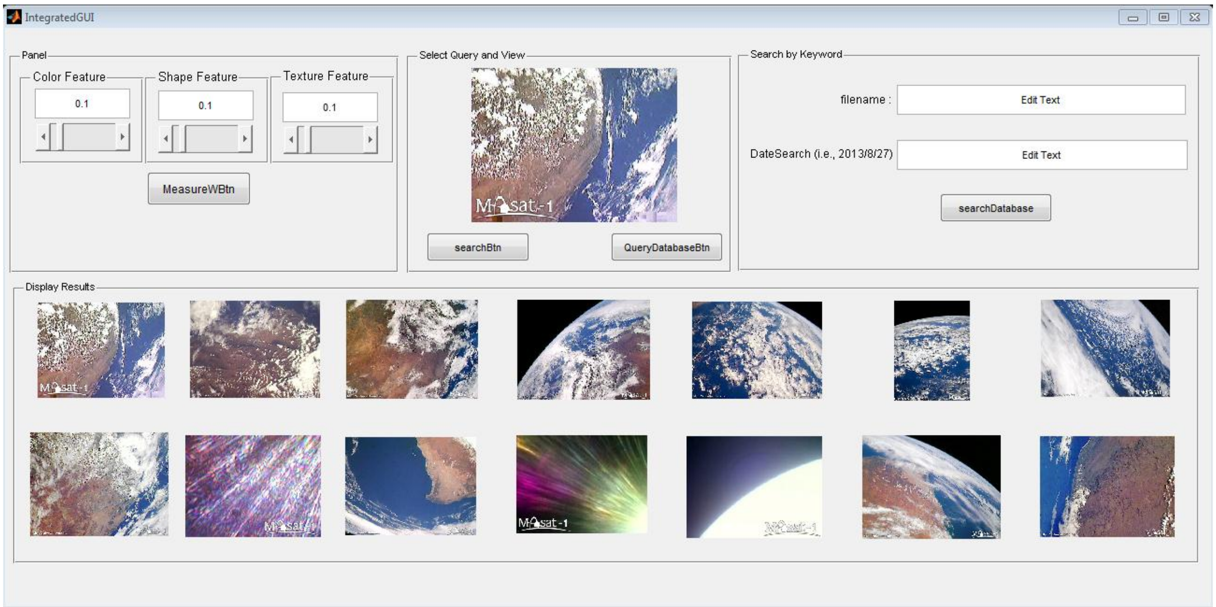


Figure 6.25: Combined result for all Features

In Figure 6.26, the date classifier results are provided. A database connection was established so that a query image's feature metadata of date and time could be extracted and compared to the database. The results shows that similar images being taken in the same year and month can be successfully retrieved.

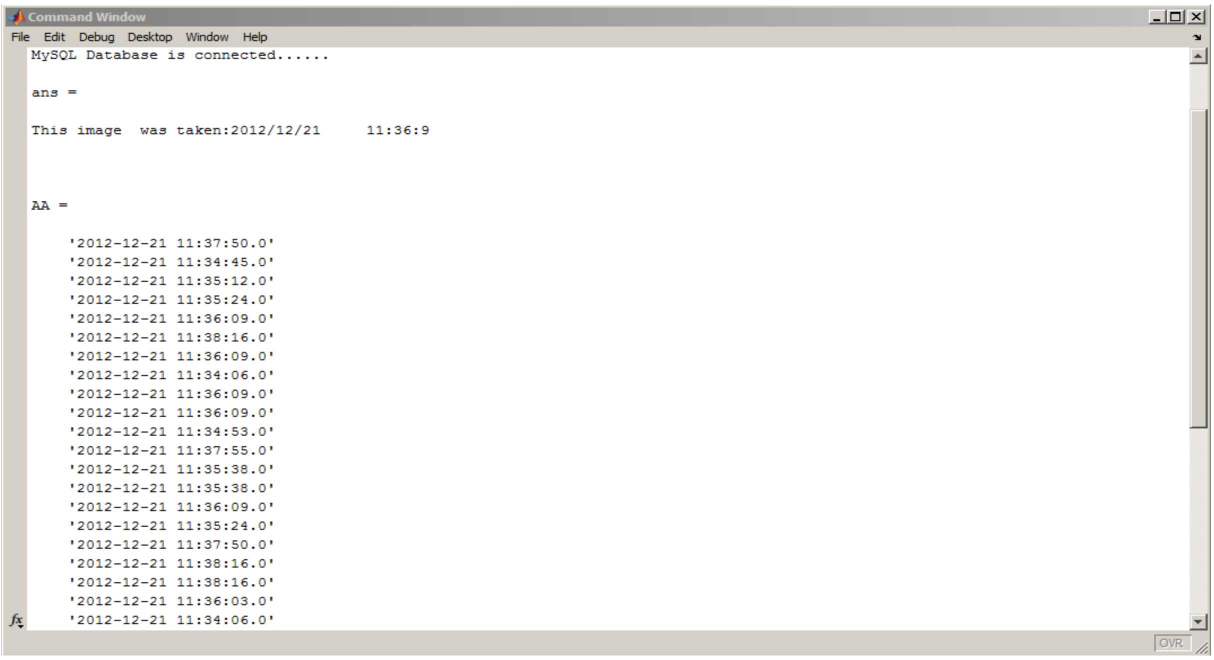


Figure 6.26: Date Classify Result

6.6 System performance and Evaluation

In content based image retrieval systems (CBIR), performance efficiency concerns the time it takes to answer the query, and the effectiveness of retrieval, which is the ability to get relevant items corresponding to the query. This is still is a major concern in CBIR systems. In a retrieval based system, queries are done against stored items in the database and distance measures are used to determine their performances, unlike matching techniques used in traditional database systems.

The performance of a CBIR system is evaluated by standard measures such as precision and recall. Precision is the ratio of the number of those relevant items against all retrieved items. Usually, precision determines the accuracy of the retrieval based system and is computed using equation 6.1, whereas recall is the ratio of those relevant retrieved items against of all stored relevant items in the database, and is a measure of robustness of the retrieval system. It is computed using equation 6.2.

$$\text{Precision} = \frac{\text{Number of relevant images retrieved}}{\text{Total number of images retrieved}} \qquad (6.1)$$

$$\text{Recall} = \frac{\text{Number of relevant images retrieved}}{\text{Total number of relevant images in the database}} \qquad (6.2)$$

6.6.1 System evaluation

The system performance is evaluated in terms of retrieval effectiveness and retrieval efficiency using precision and recall. To evaluate the proposed system against other existing systems, a commonly available image database (Wang) is used. It is an image database which is a part of the Corel database of one thousand images that are grouped into ten classes (African, Beaches, Buildings, Buses, Dinosaurs, Elephants, Flowers, Horses, Mountains and Foods). Each class contains one hundred images (see Appendix B). It has a diverse range of images to assist in evaluating classification of the proposed system. This image database is used for testing purposes by most of the CBIR systems. Although this project focuses on CubeSat satellite images, there is no standard image database available to compare the proposed system against other systems.

To test and evaluate the proposed system, four algorithms were designed, namely colour histogram, texture, shape and date classifier. For each of these algorithms, the retrieval effectiveness and efficiency were verified. To test the proposed system, thirty images from each class were randomly selected, thus reducing the available one hundred images from each class to thirty. A total of three hundred images were used in ten groups of thirty images each. For each class, a sample image (query) were imported into the Matlab workspace to allow to find similar images in the database and a maximum of fourteen images similar to the query were retrieved. It should be noted that for testing purposes the maximum of fourteen images were chosen to adequately show the results on-screen. The algorithms allow for all similar images to be displayed on-screen.

A. Color Histogram algorithm evaluation

In the first evaluation, from the African class, for each image the features were extracted and stored within the database. In Figure 6.27 a query image from the African people class was selected. Fourteen similar images belonging to the same class were subsequently retrieved. All the images retrieved by the proposed system are relevant, as per the defined feature to the query image. The colour histogram algorithm used to retrieve images thus ensures that images consisting of the same content belong to one class. It should be noted that the purpose of the proposed system is to retract similar images that belong to a specific group and that contain specific features. The purpose is not merely to retrieve an exact match of the query against images in the database.

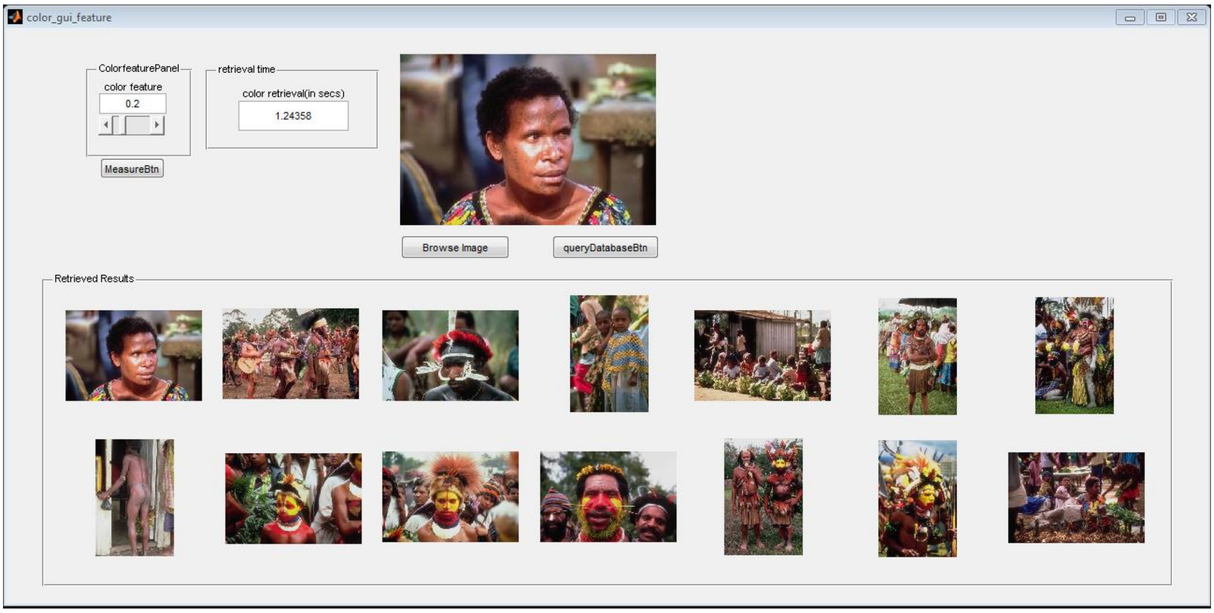


Figure 6.27: Colour histogram colour evaluation: African class

The second evaluation focused on the Beaches image class. Thirty random images were selected and, for each, their features extracted and stored in the database. As seen in Figure 6.28, a query image was selected from the beaches class to serve as input. The fourteen retrieved images belong to the same class with the requested features as the query image. Therefore, the precision of the colour histogram algorithm is 1 (14/14), meaning all images are relevant to the query image.

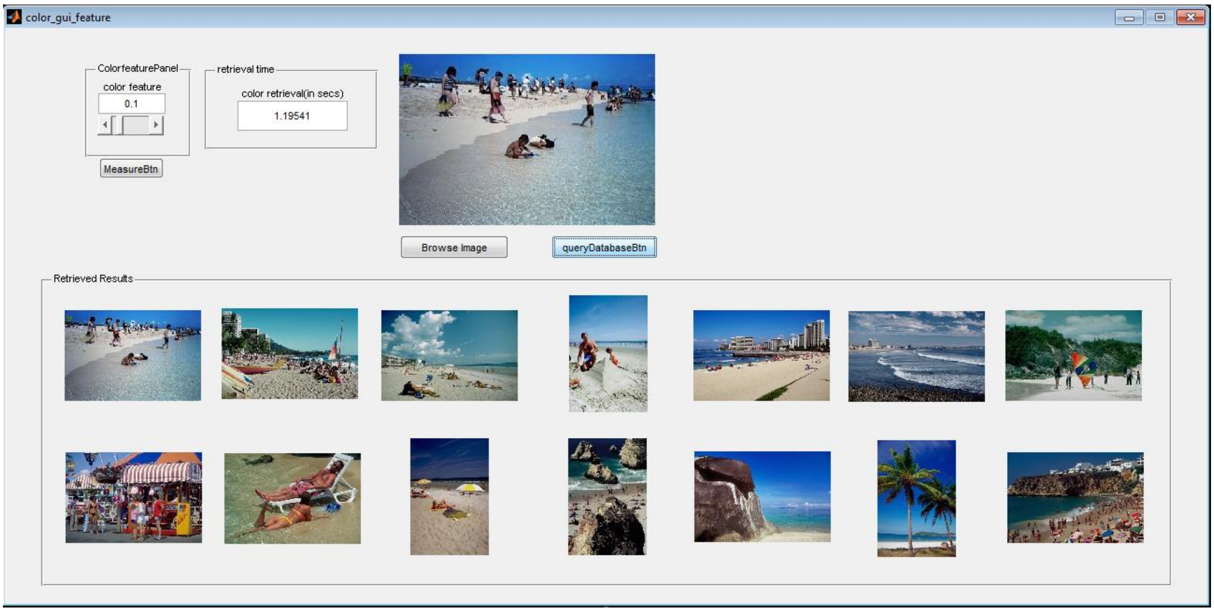


Figure 6.28: Colour histogram colour evaluation: Beach class

The third evaluation is from the Buildings image class, where thirty random images were also selected. Image features were extracted and as shown in Figure 6.29, a query image was selected. The fourteen output images belong to the same class as the query image. The precision of the colour histogram algorithm is thus 1 (14/14).

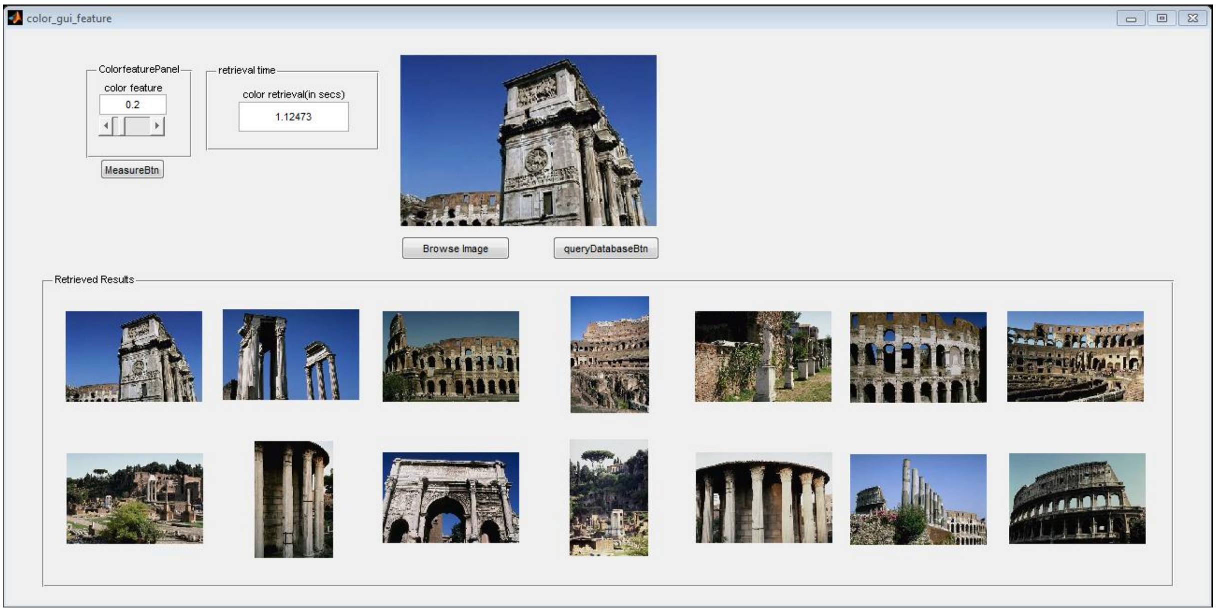


Figure 6.29: Colour histogram colour evaluation: Building class

The fourth evaluation focuses on the Buses image class, where thirty random images were also selected. As shown in the Figure 6.30, all retrieved results belong to the same class as the query image. The achieved precision of the colour histogram algorithm from this class is 1 (14/14).

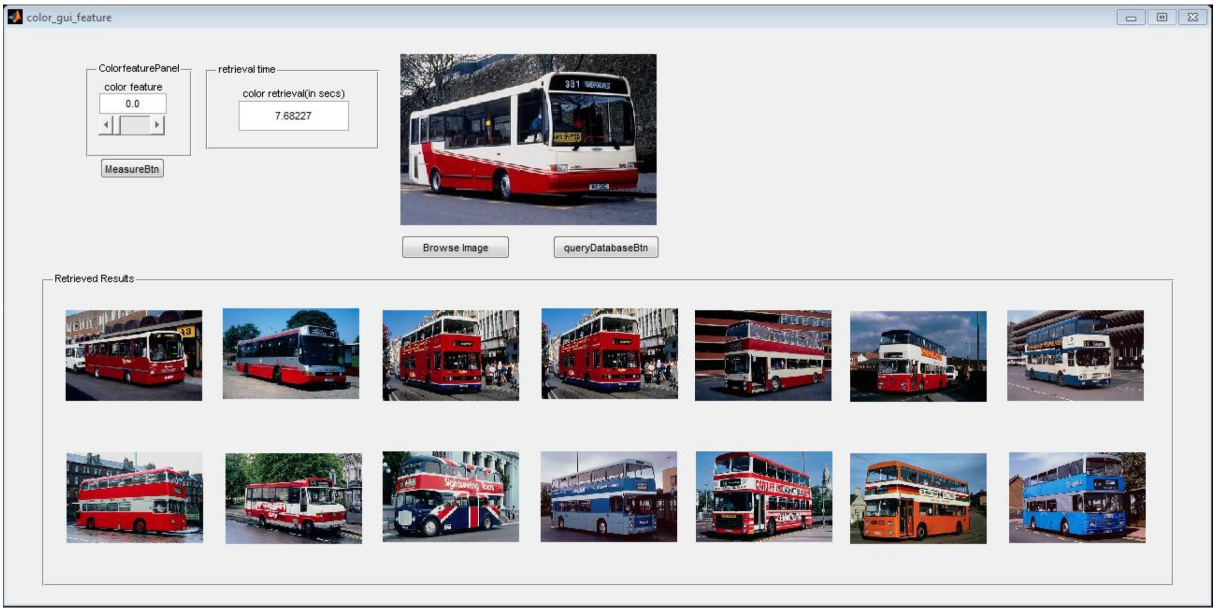


Figure 6.30: Colour histogram colour evaluation: Buses class

Figure 6.31 shows the dinosaurs image class that was used within the fifth evaluation. The thirty random images led to the retrieval of fourteen images. Out of the fourteen one image from the bus class was also retrieved. In this instance, the bus image has the same quantity colour as that of images within the dinosaur class. The precision is therefore 0.928.

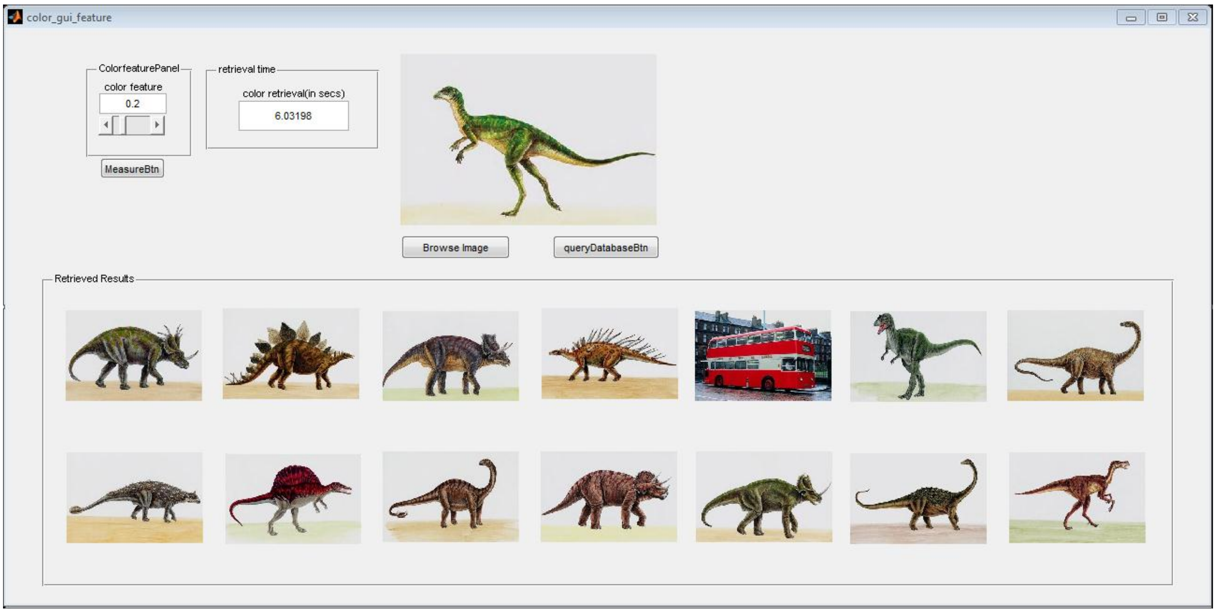


Figure 6.31: Colour histogram colour evaluation: Dinosaur class

Figure 6.32 shows the details of the sixth evaluation where images from the Elephant class were used. The precision of the colour histogram algorithm is 1 (14/14).

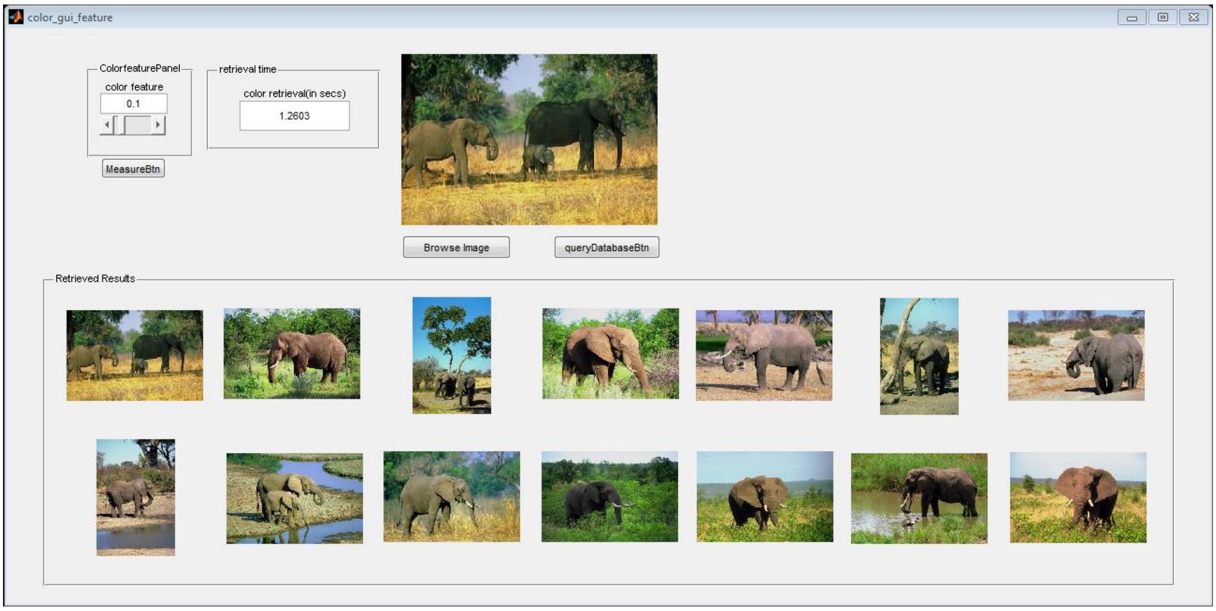


Figure 6.32: Colour histogram colour evaluation: Elephant class

The flowers image class was used within the seventh evaluation and the results retrieved are shown in Figure 6.33. All retrieved images belonged to the same class as the query image. Therefore, the precision of the colour histogram algorithm is 1 (14/14) with a recall figure of 0.4667 (14/30). The same results of 1 (14 of 14 images) were obtained when tested on the Horses class (see Figure 6.34), the Mountains class (see Figure 6.35) and the Food class (Figure 6.36).

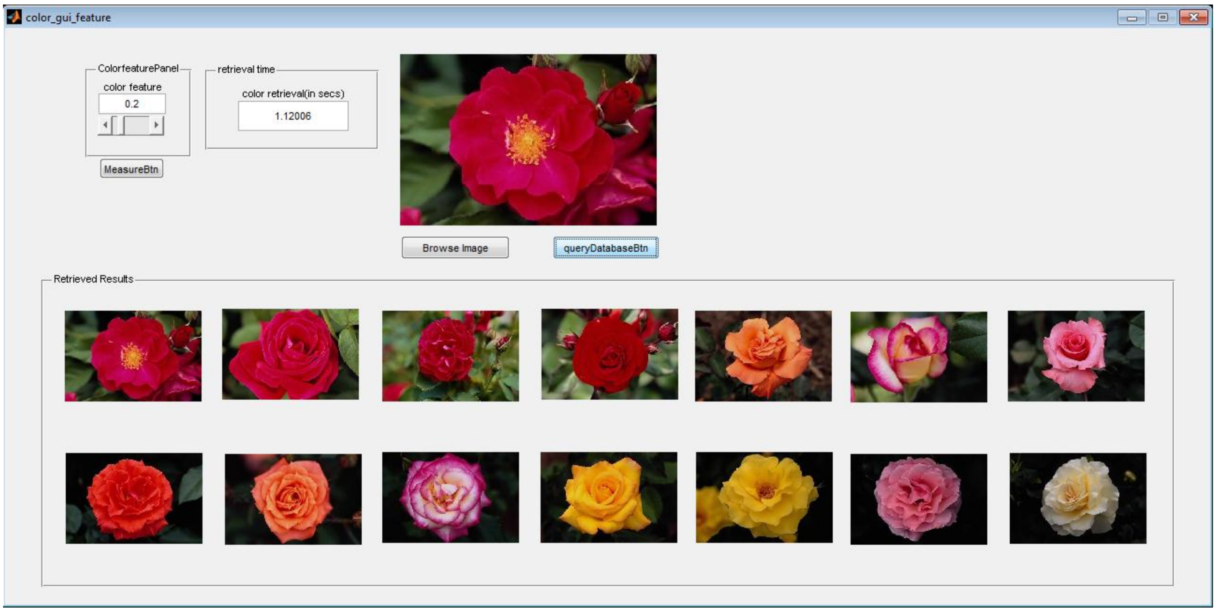


Figure 6.33: Colour histogram colour evaluation: Flower class

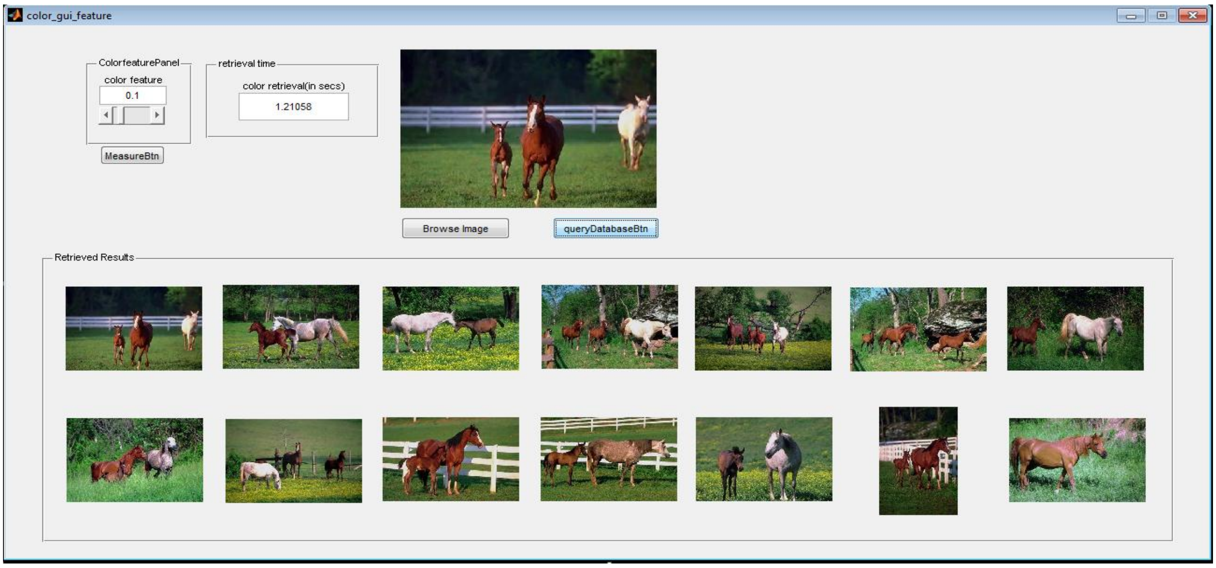


Figure 6.34: Horse query, the fourteen similar retrieved images

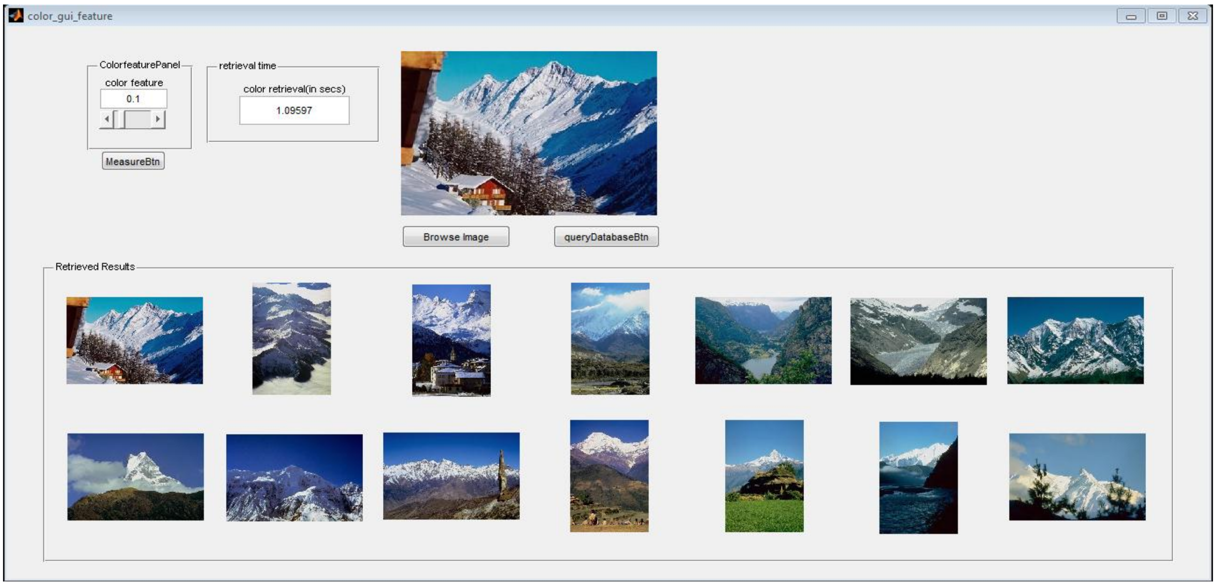


Figure 6.35: Colour histogram colour evaluation: Mountain class

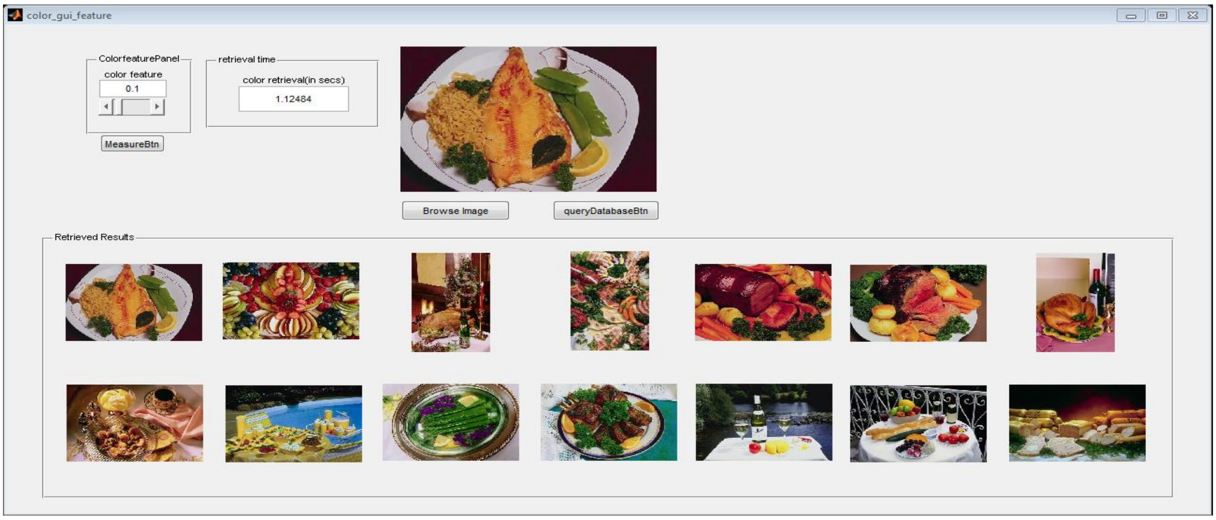


Figure 6.36: Colour histogram colour evaluation: Food query

Figure 6.37 shows Ikonos and Quick bird images used to evaluate the algorithm's effectiveness and efficiency. These images are of high resolution unlike tests done on the previous ten classes. The results achieved indicate an efficiency of 1 (14/14).

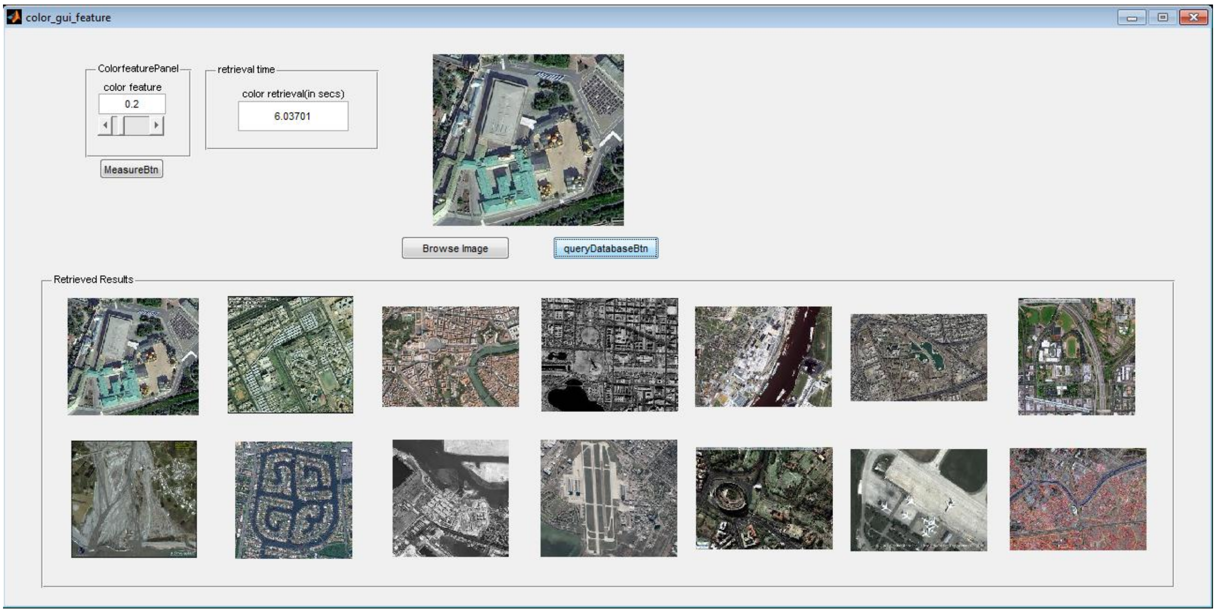


Figure 6.37: Colour histogram colour evaluation: Ikonos class

B. GLCM algorithm evaluation

The texture GLCM algorithm extracts information embedded in an image and classifies them into classes in order to estimate the similarity between two images. The four statistical features, namely energy, contrast, homogeneity and correlation, are used to test the efficiency and correctness of the proposed design. Using a query image of a building, the four texture properties are extracted and compared to the images in the database.

The texture feature algorithm (GLCM algorithm) is evaluated based on the Wang database of randomly selected images from thirty images in each class. Following this same testing and evaluation process, results of 1 were obtained in all ten test classes, namely the African class (Figure 6.38), Beach class (Figure 6.39), Building class (Figure 6.40), Busses class (Figure 6.41), Dinosour class (Figure 6.42), Elephant class (Figure 6.43), Flower class (Figure 6.44), Horse class (Figure 6.45), Mountain class (Figure 6.46), Food class (Figure 6.47) and Masat-1 (Figure 6.48). Evaluations within all of these classes reflected precision values of 1.

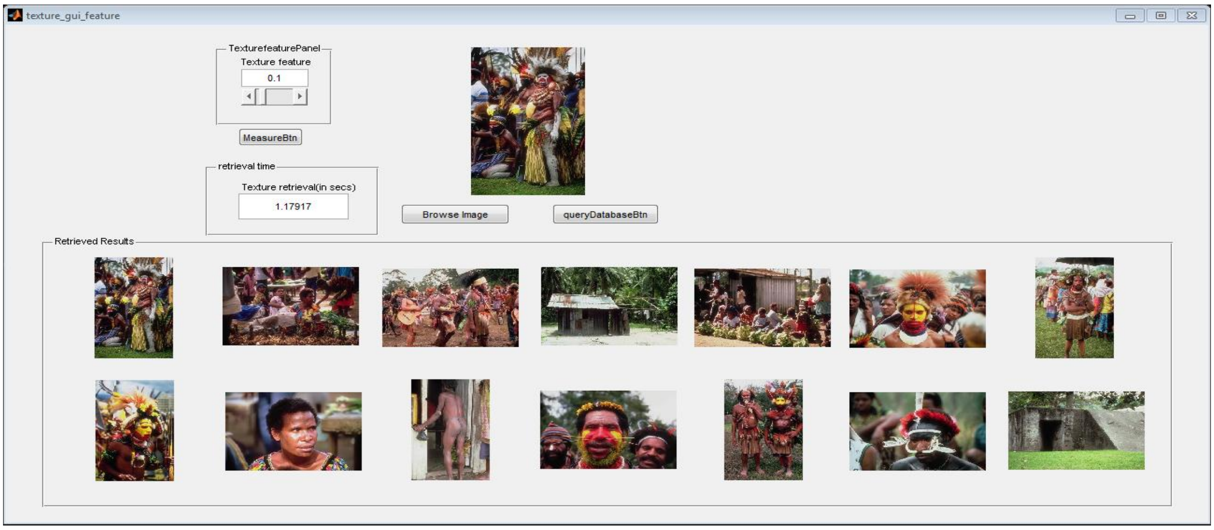


Figure 6.38: GLCM algorithm evaluation: African class

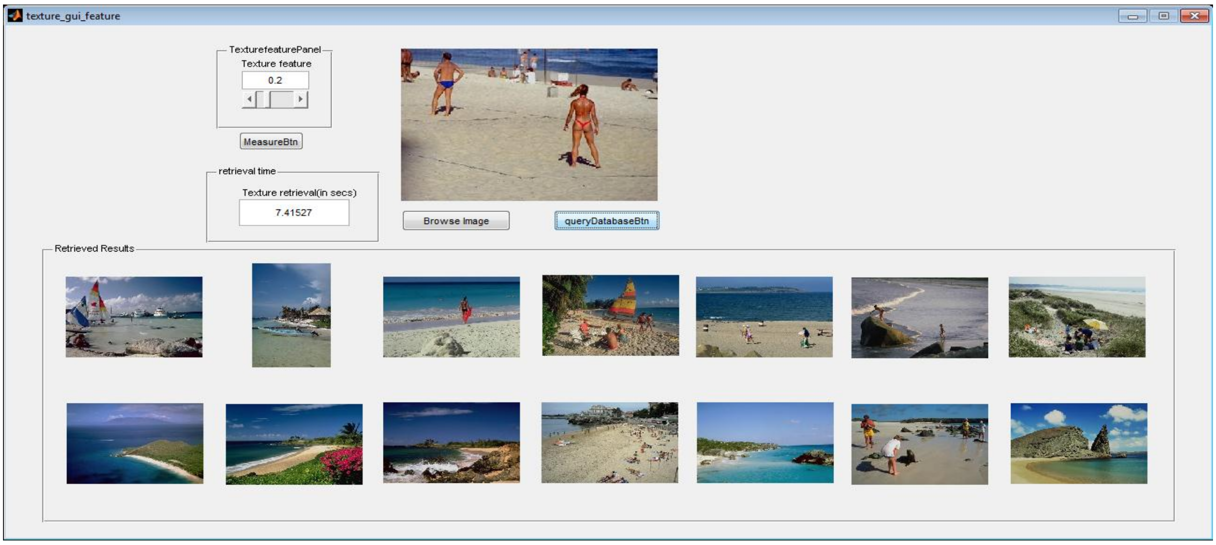


Figure 6.39: GLCM algorithm evaluation: Beach class

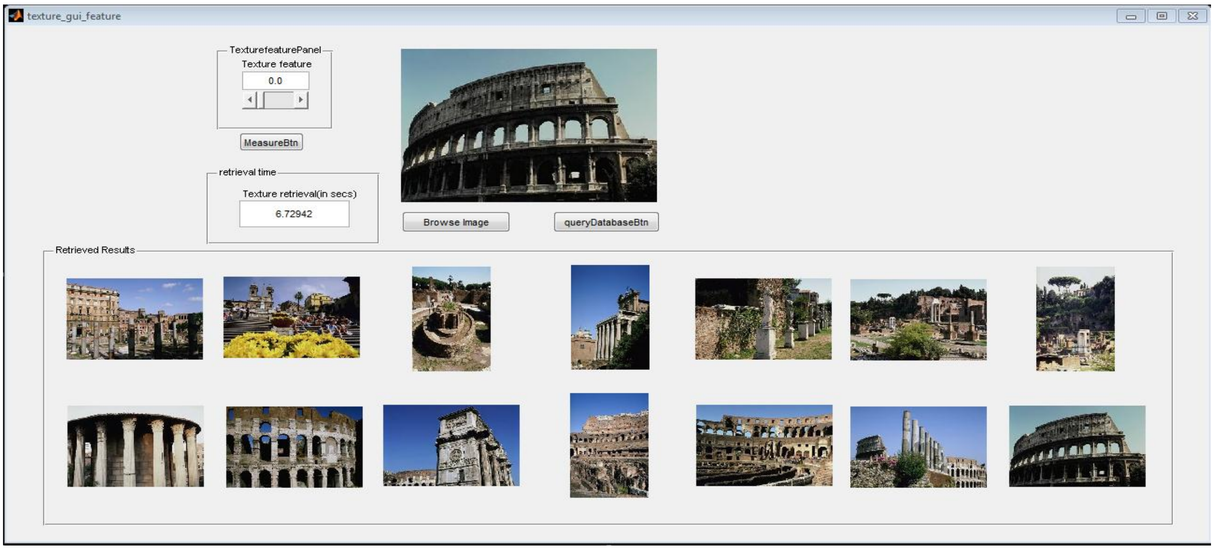


Figure 6.40: GLCM algorithm evaluation: Building class

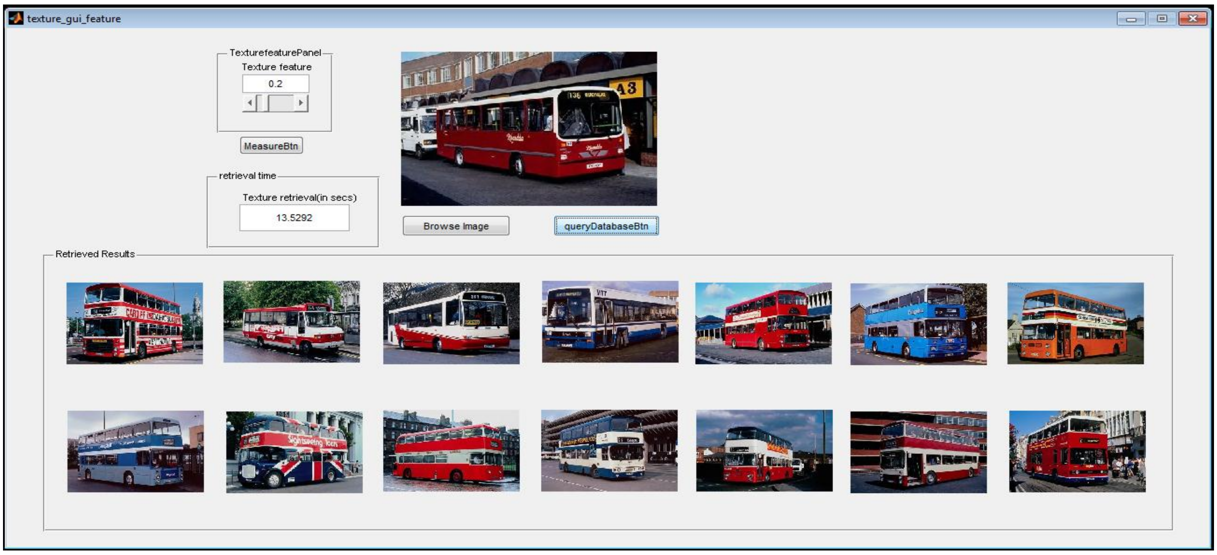


Figure 6.41: GLCM algorithm evaluation: Bus class

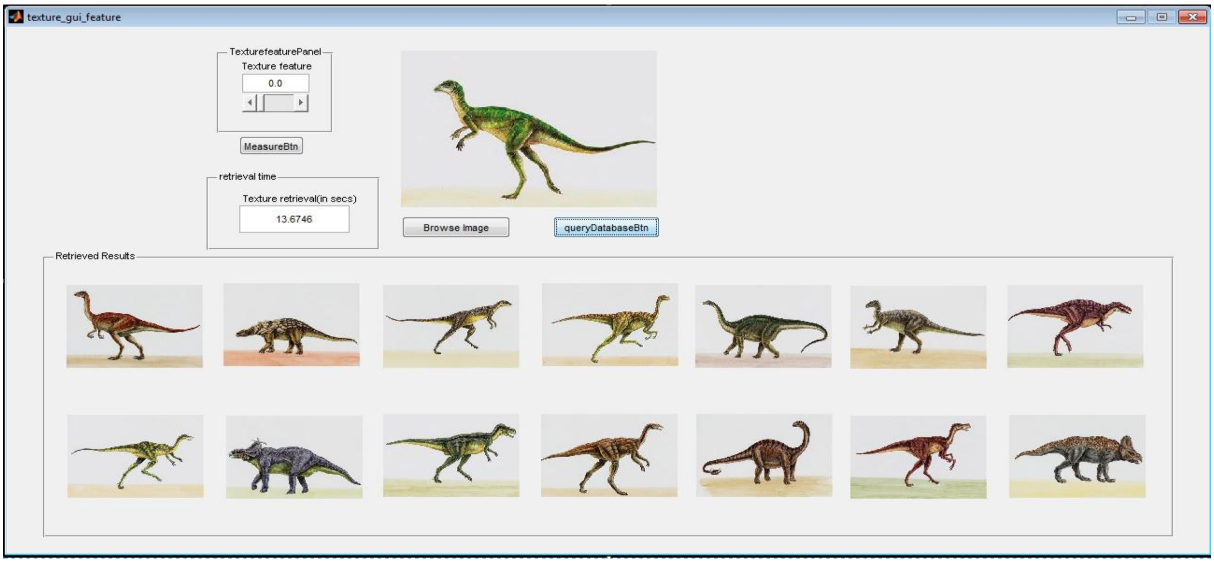


Figure 6.42: GLCM algorithm evaluation: Dinosaur class

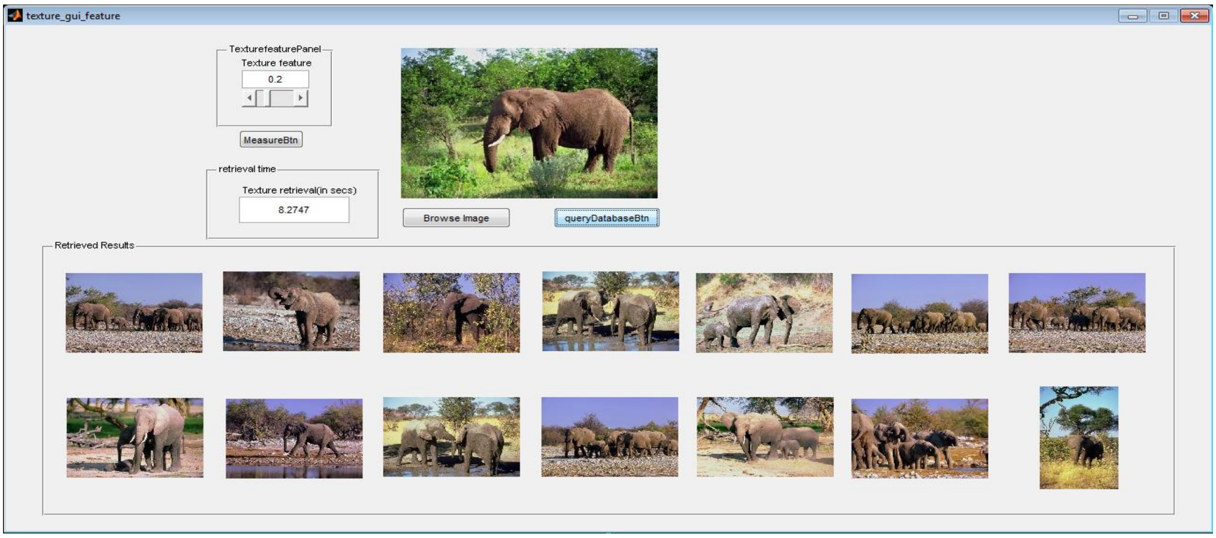


Figure 6.43: GLCM algorithm evaluation: Elephant class

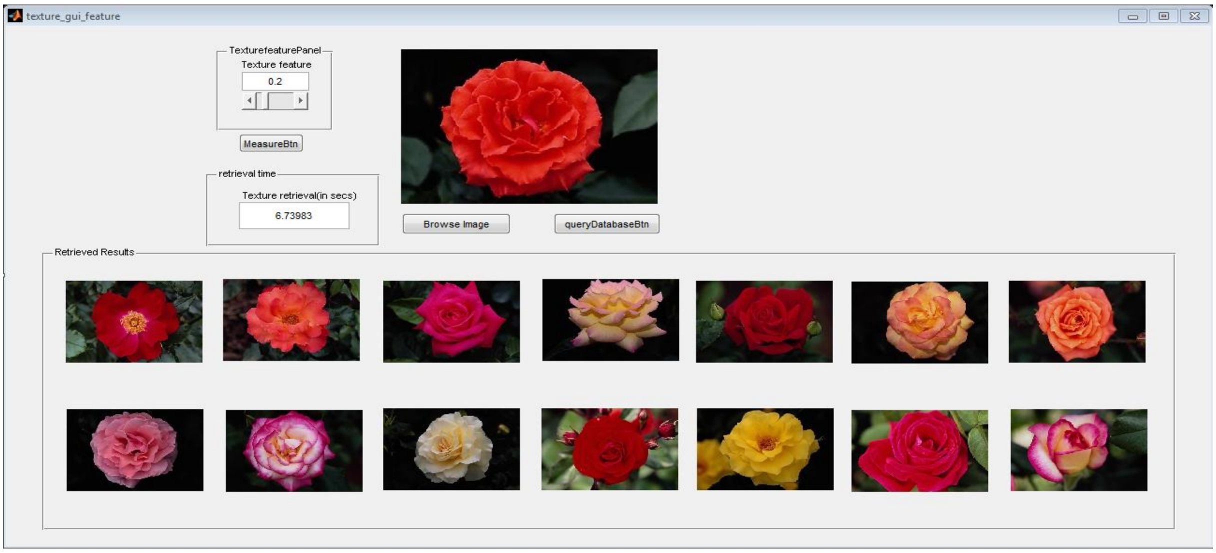


Figure 6.44: GLCM algorithm evaluation: Flower class

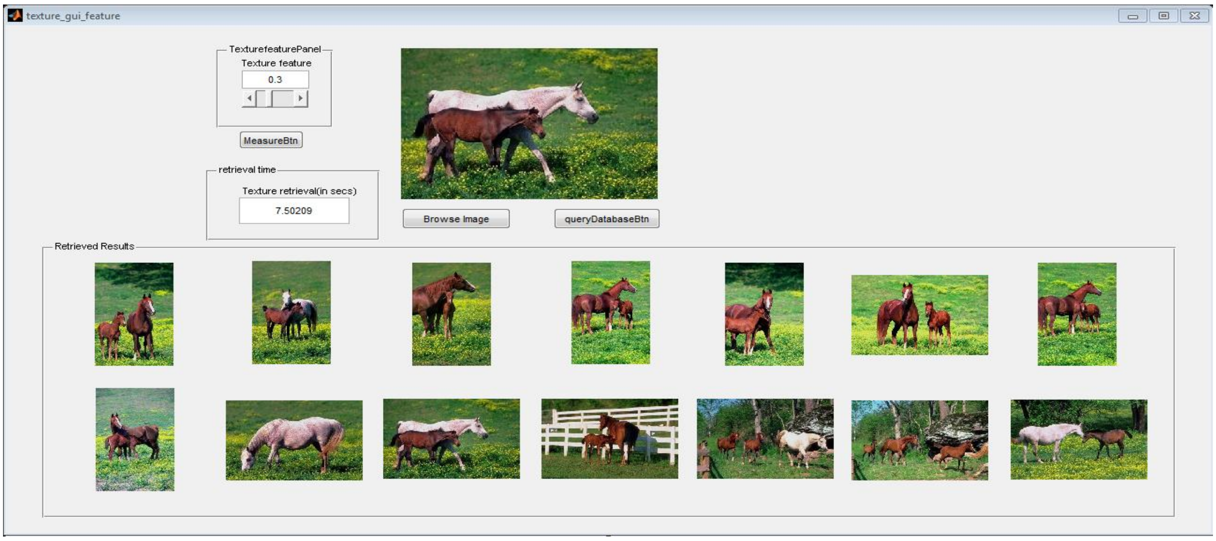


Figure 6.45: GLCM algorithm evaluation: Horse class

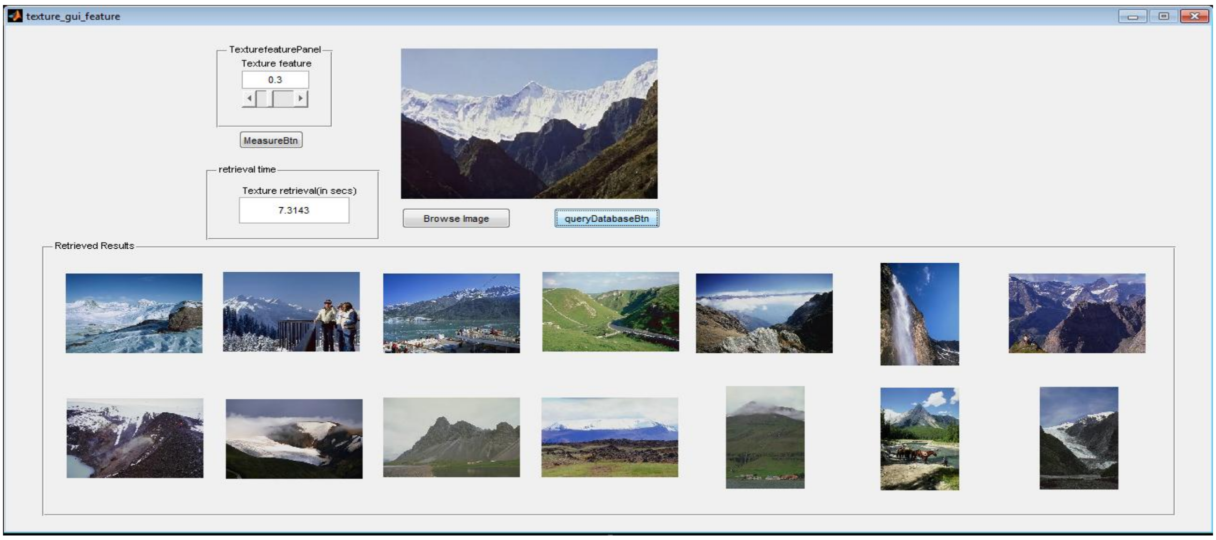


Figure 6.46: GLCM algorithm evaluation: Mountain class

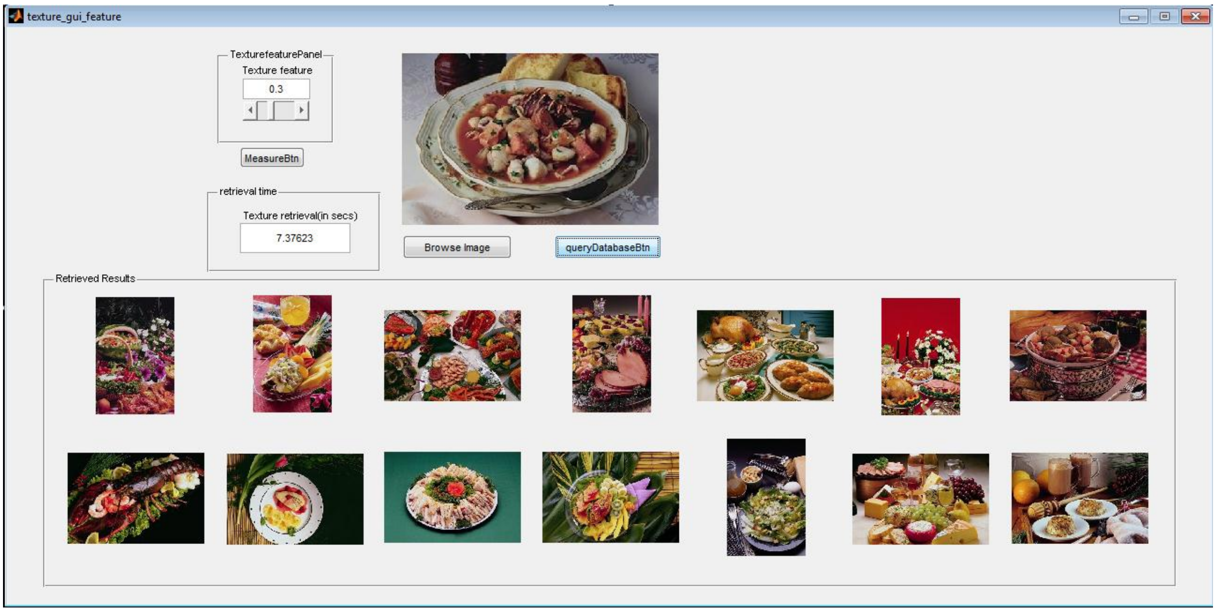


Figure 6.47: GLCM algorithm evaluation: Food class

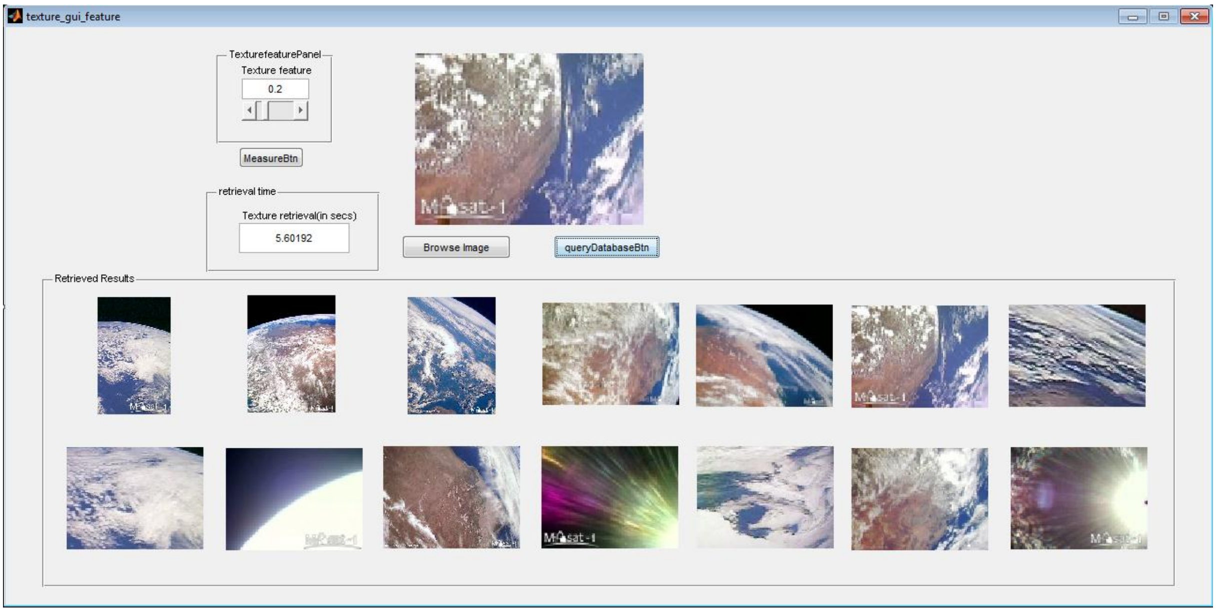


Figure 6.48: GLCM algorithm evaluation: Masat-1 class

C. Mathematical Morphology algorithm evaluation

The shape feature algorithm (Mathematical Morphology algorithm) is evaluated based on the Wang database of thirty randomly selected images from each class. The images' features are extracted and stored into the database. The shape mathematical morphology algorithm is evaluated through queries within different image classes to find similar images in the database. To accommodate space restrictions only fourteen relevant images are shown. The following number of figures shows the results and depicts the fourteen retrieved images from the African class (Figure 6.49), Beach class (Figure 6.50), Building class (Figure 6.51), Bus class (Figure 6.52), Dinosaur class (Figure 6.53), Elephant class (Figure 6.54), Flower class

(Figure 6.55), Horse class (Figure 6.56), Mountain class (Figure 6.57) and Food class (Figure 6.58). The results obtained indicate a precision of 1 within all of the queries.

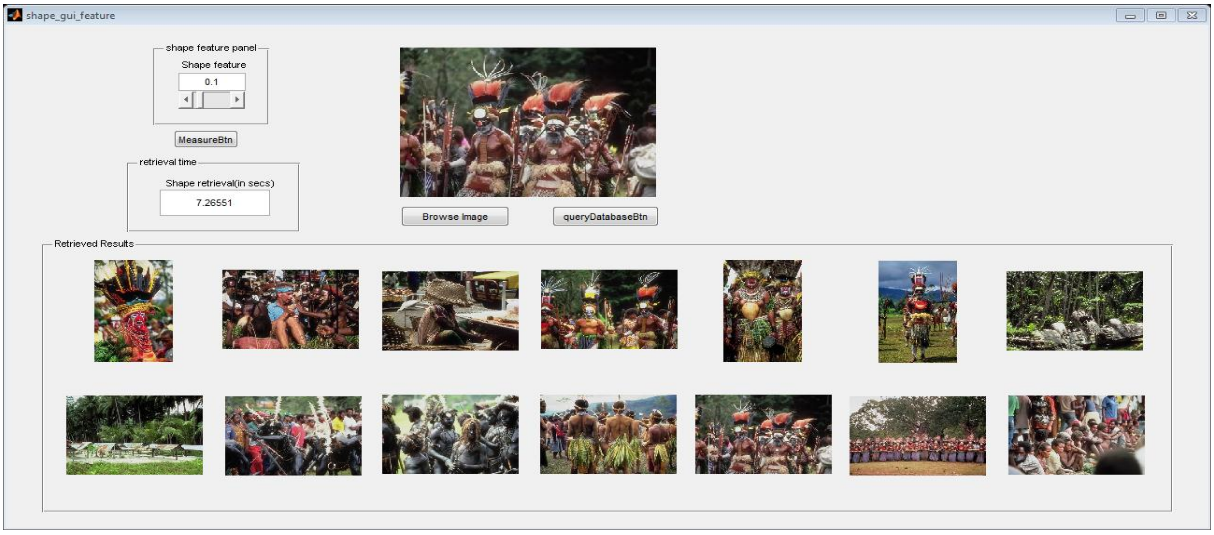


Figure 6.49: Mathematical Morphology algorithm evaluation: African class

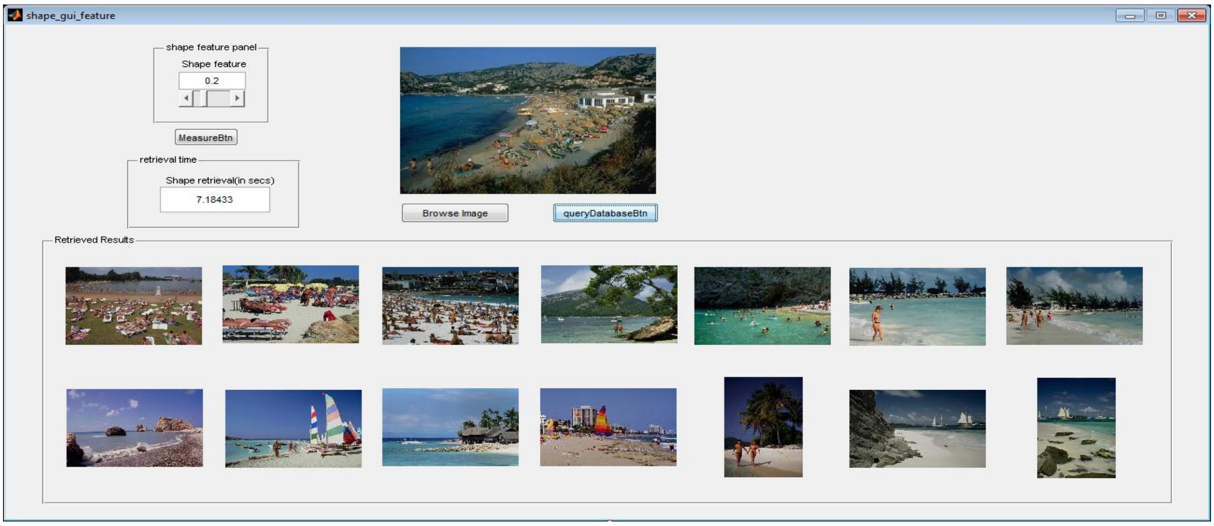


Figure 6.50: Mathematical Morphology algorithm evaluation: Beach class

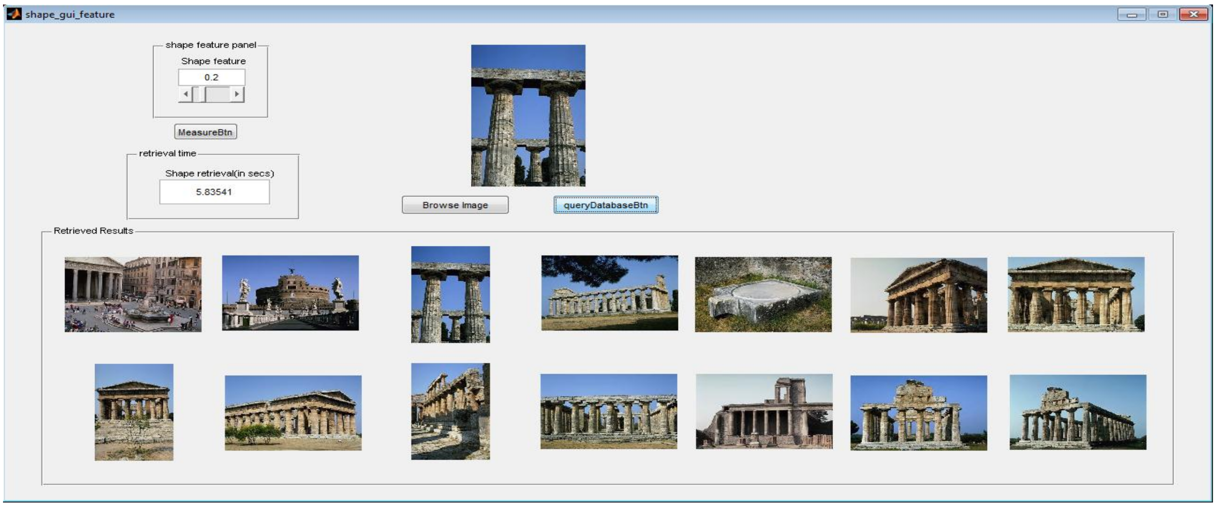


Figure 6.51: Mathematical Morphology algorithm evaluation: Building class

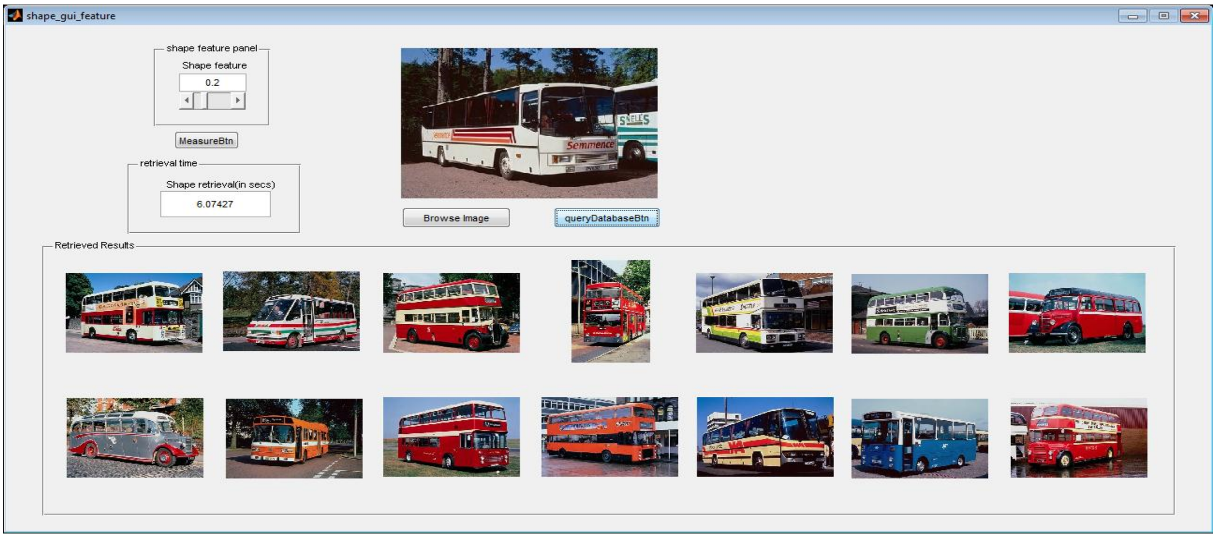


Figure 6.52: Mathematical Morphology algorithm evaluation: Bus class

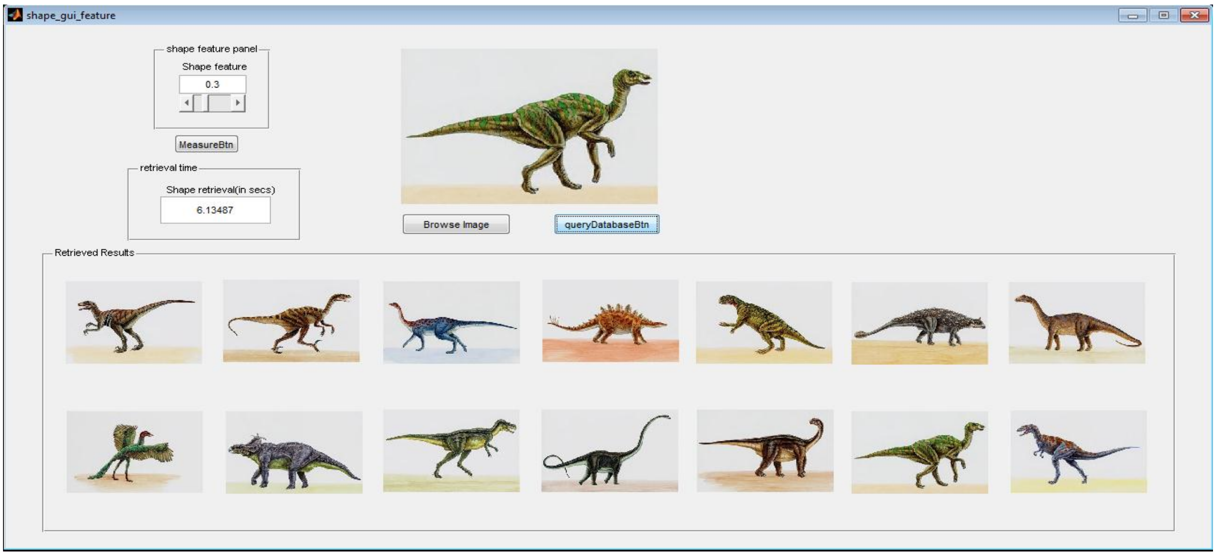


Figure 6.53: Mathematical Morphology algorithm evaluation: Dinasaur class

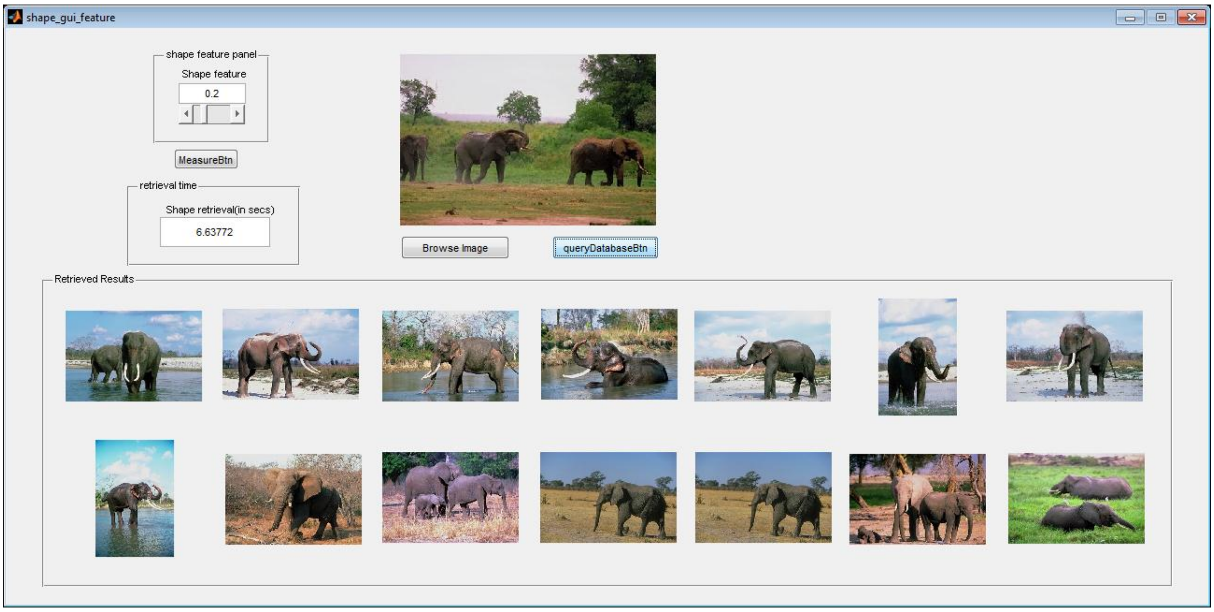


Figure 6.54: Mathematical Morphology algorithm evaluation: Elephant class

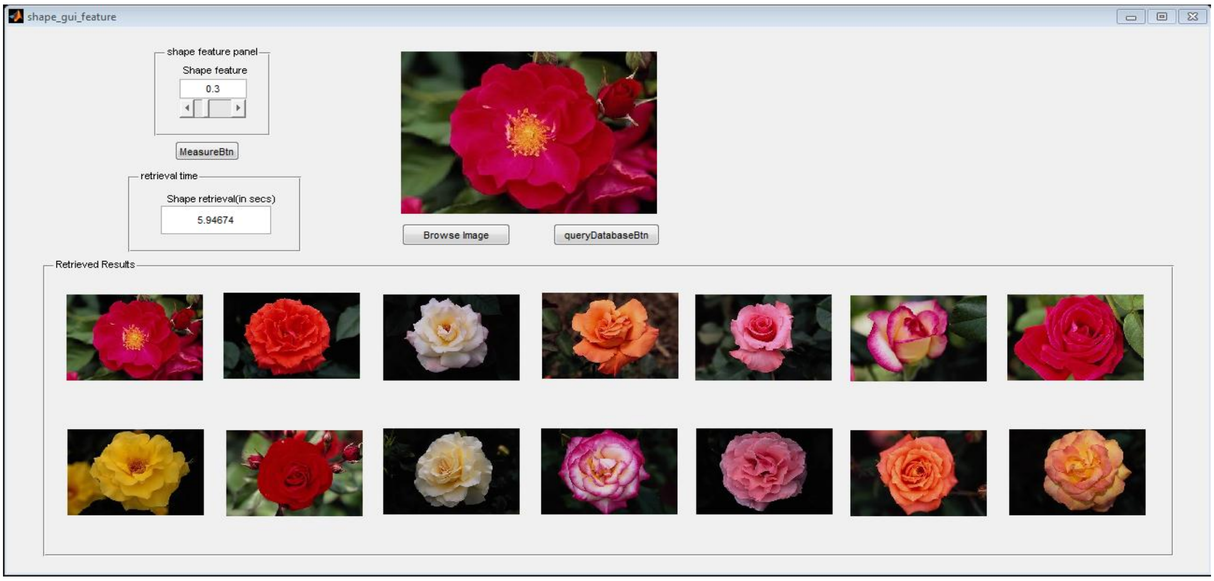


Figure 6.55: Mathematical Morphology algorithm evaluation: Flower class

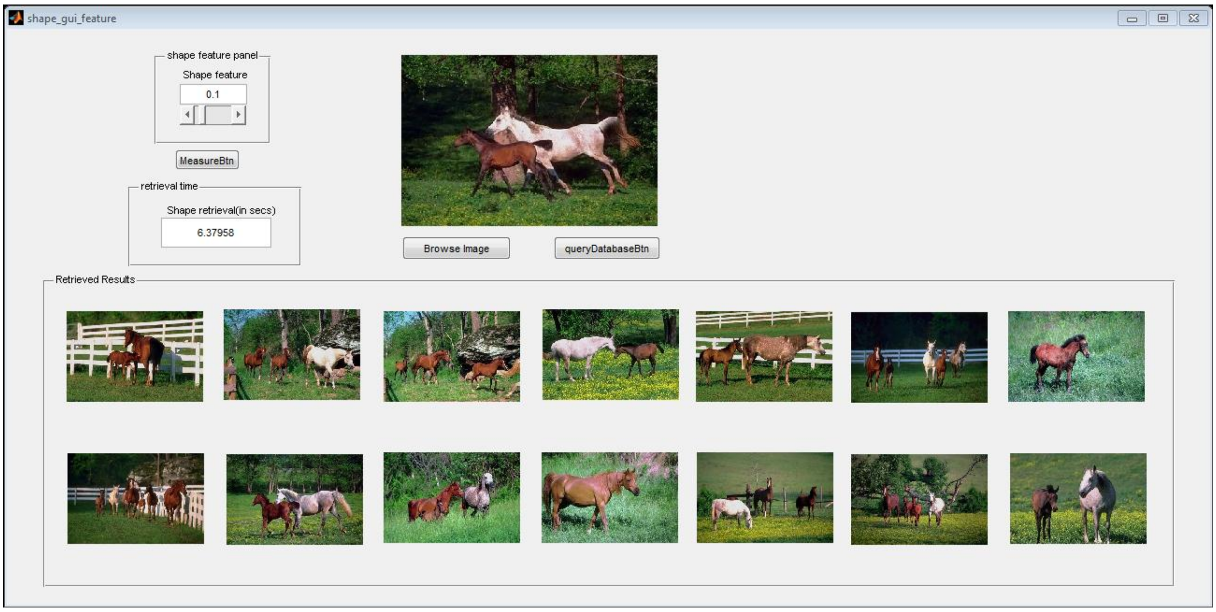


Figure 6.56: Mathematical Morphology algorithm evaluation: Horses class

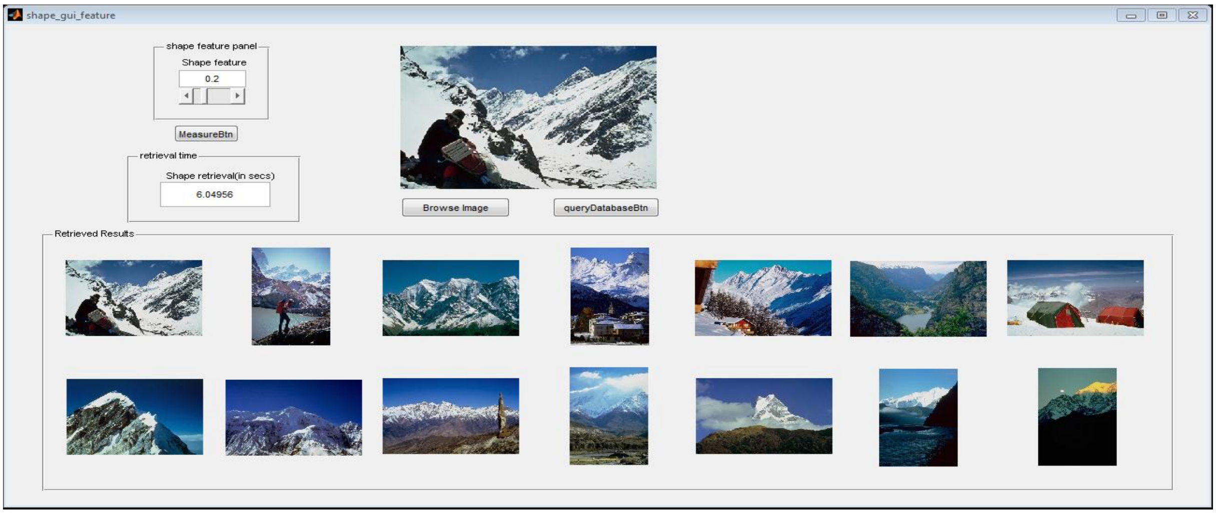


Figure 6.57: Mathematical Morphology algorithm evaluation: Mountain class

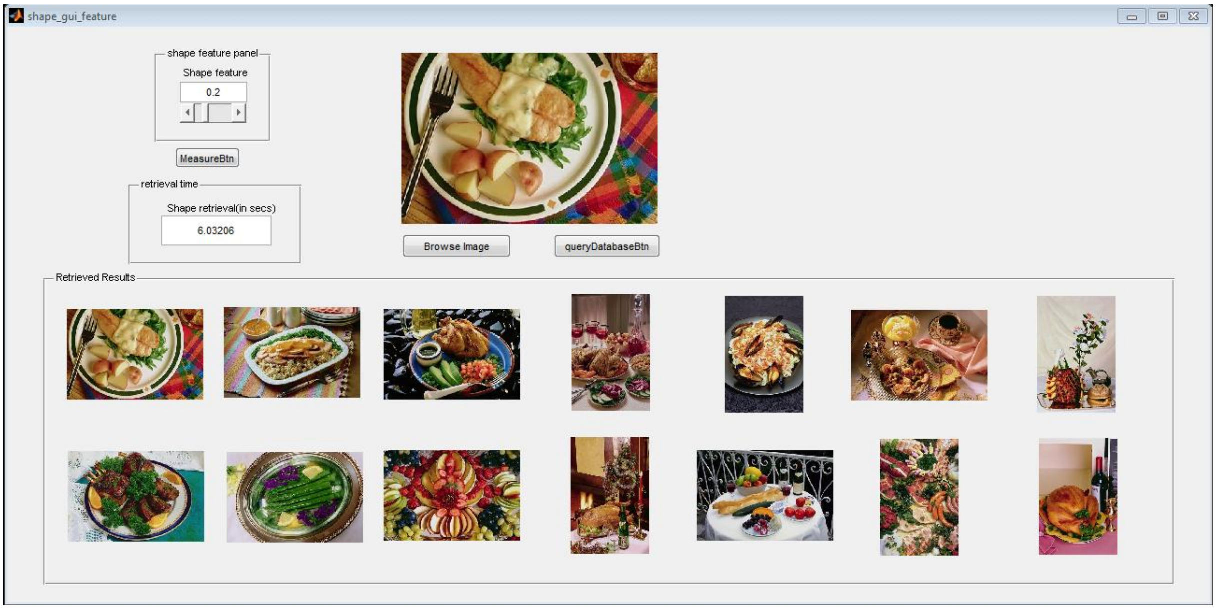


Figure 6.58: Mathematical Morphology algorithm evaluation: Food class

D. Integrated proposed system evaluation

The Integrated proposed system algorithm is evaluated based on the Wang database of thirty randomly selected images in each class. For each of the images, the features are extracted and categorized. For the proposed integrated system, a number of varying classes of images were used to test and evaluate the integrated algorithm. Different image classes and queries were selected and a precision of 1 was observed in all of the evaluations. For specifics please see retrievals from the African class (Figure 6.59), Beach class (Figure 6.60), Building class (Figure 6.61), Bus class (Figure 6.62), Dinosaur class (Figure 6.63), Elephant class (Figure 6.64), Flower class (Figure 6.65), Horse class (Figure 6.66), Mountain class (Figure 6.67) and Food class (Figure 6.68).

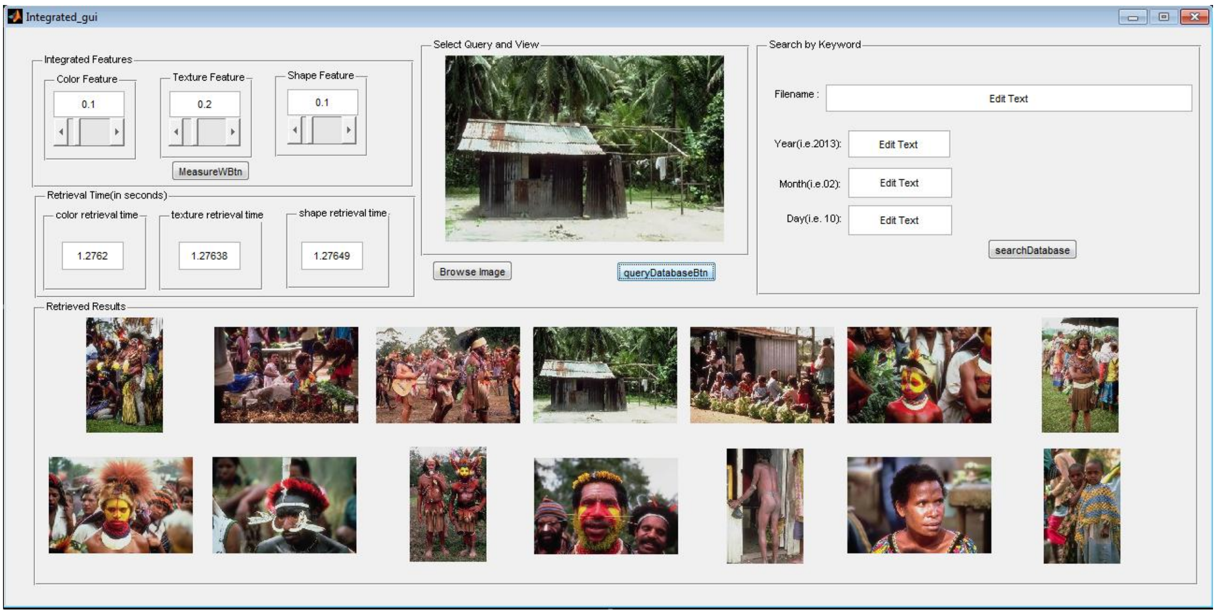


Figure 6.59: Integrated proposed system evaluation: African class

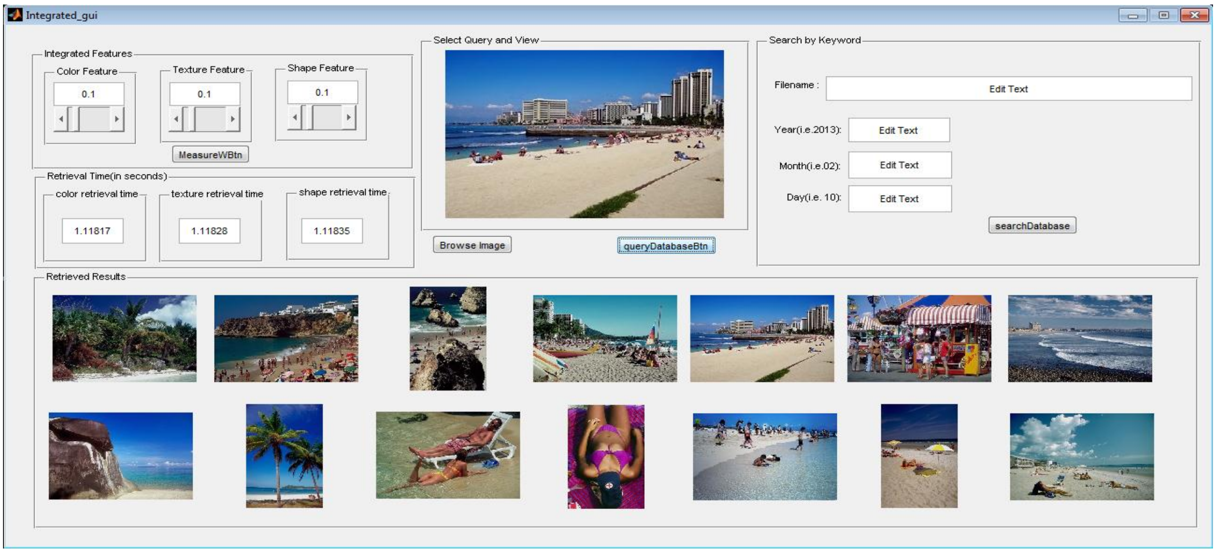


Figure 6.60: Integrated proposed system evaluation: Beach class

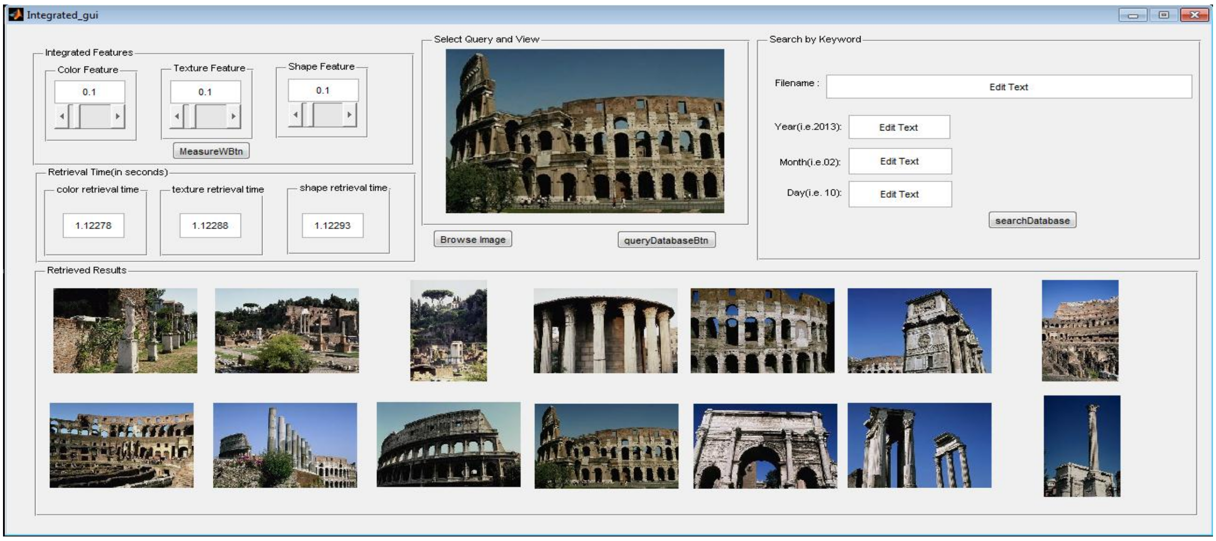


Figure 6.61: Integrated proposed system evaluation: Building class

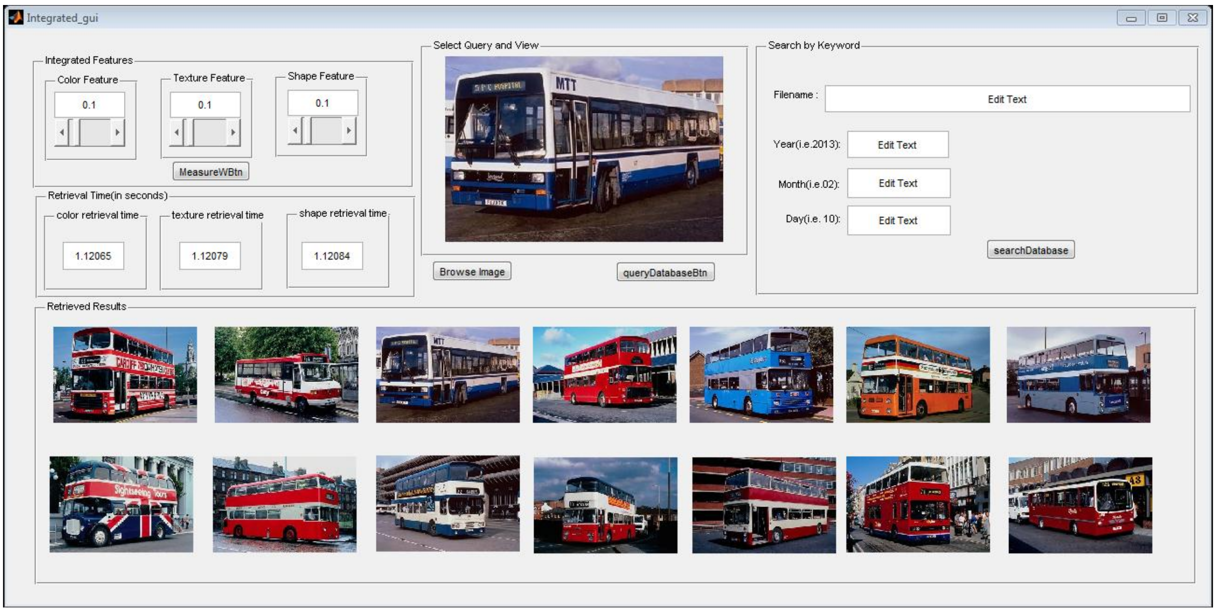


Figure 6.62: Integrated proposed system evaluation: Bus class

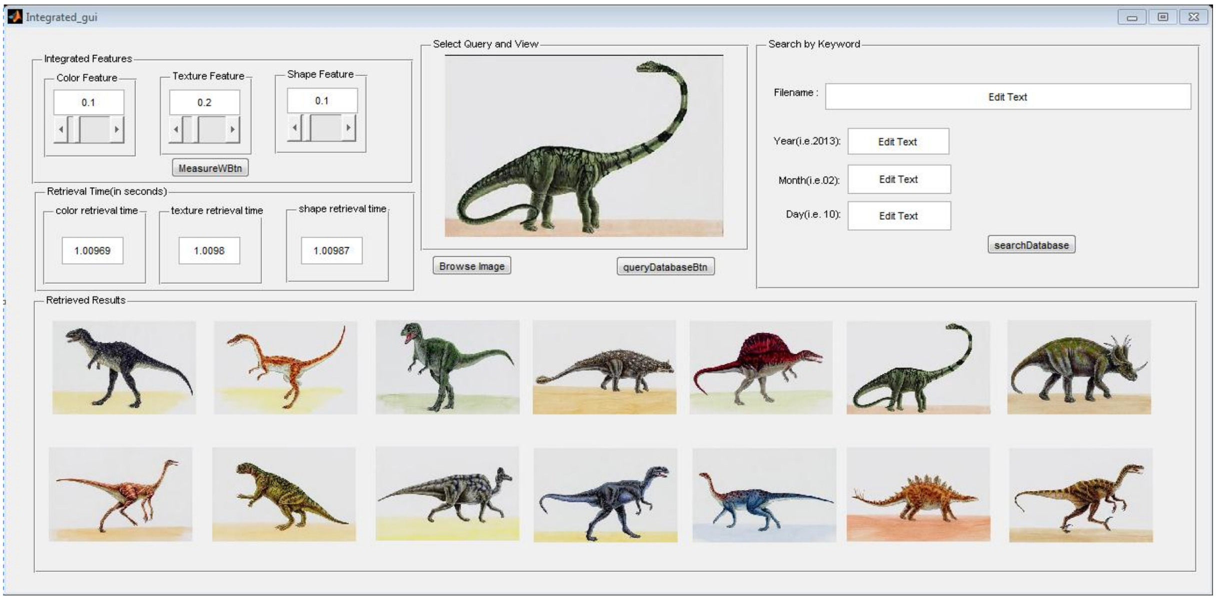


Figure 6.63: Integrated proposed system evaluation: Dinosaur class

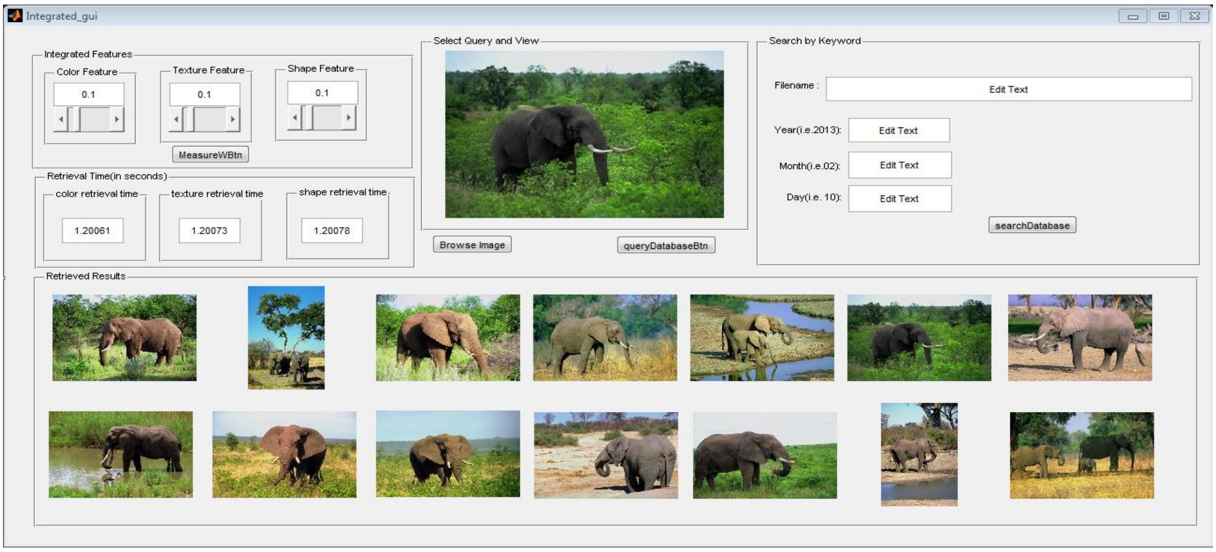


Figure 6.64: Integrated proposed system evaluation: Elephant class

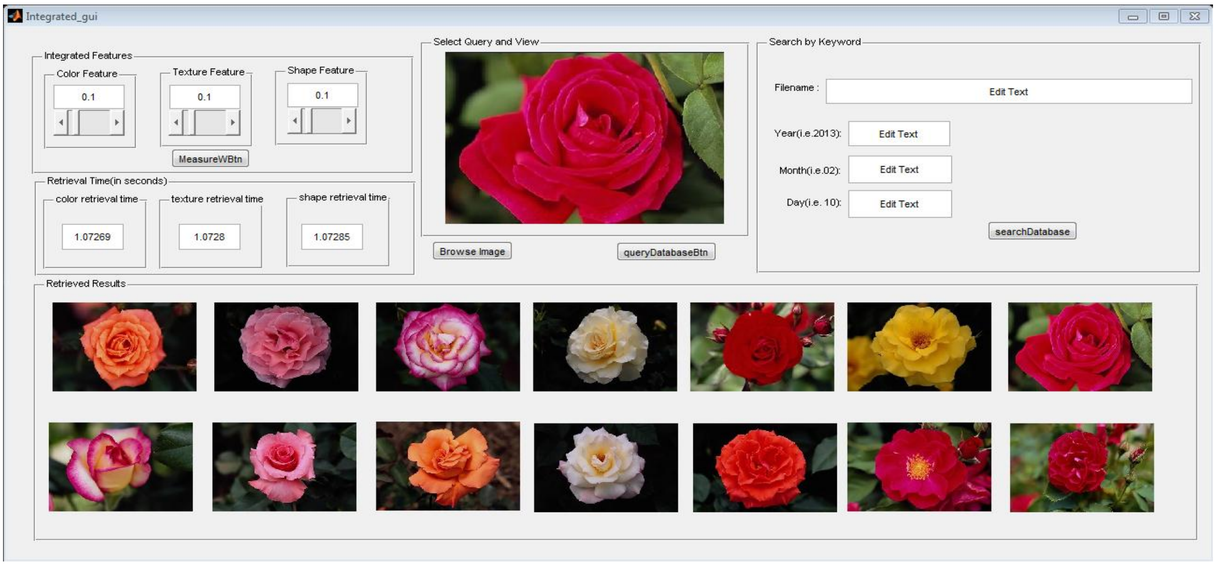


Figure 6.65: Integrated proposed system evaluation: Flower class

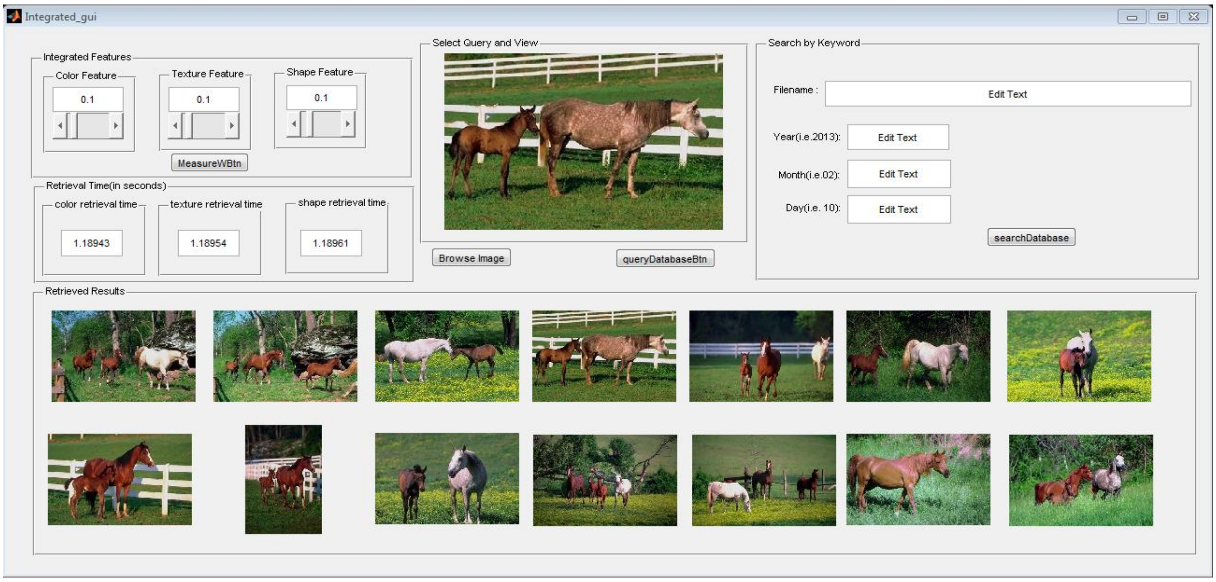


Figure 6.66: Integrated proposed system evaluation: Horse class

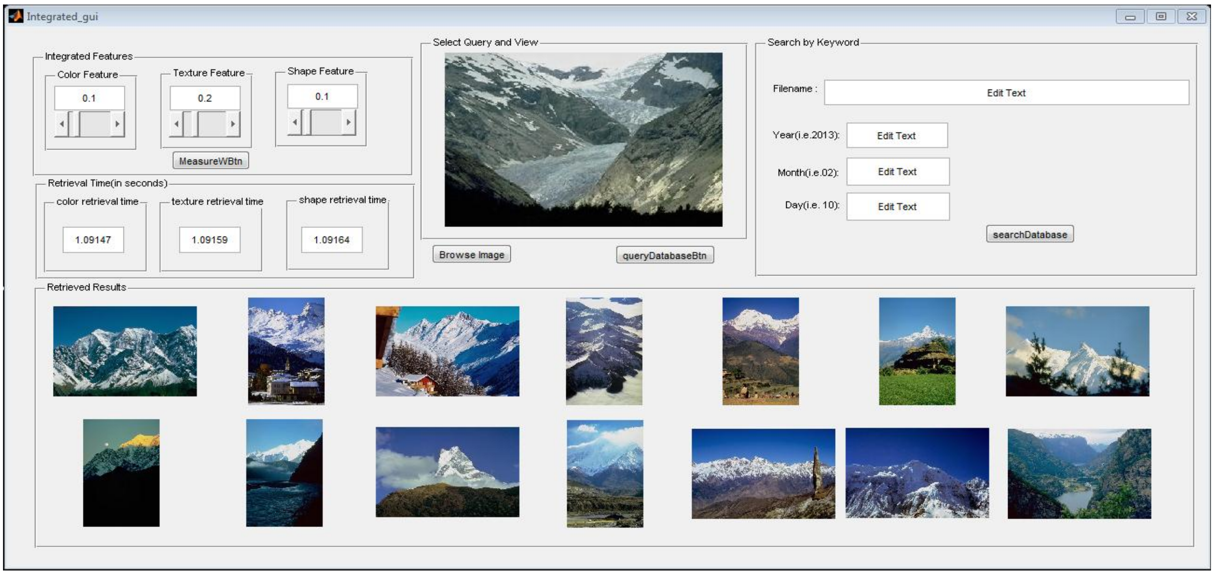


Figure 6.67: Integrated proposed system evaluation: Mountain class

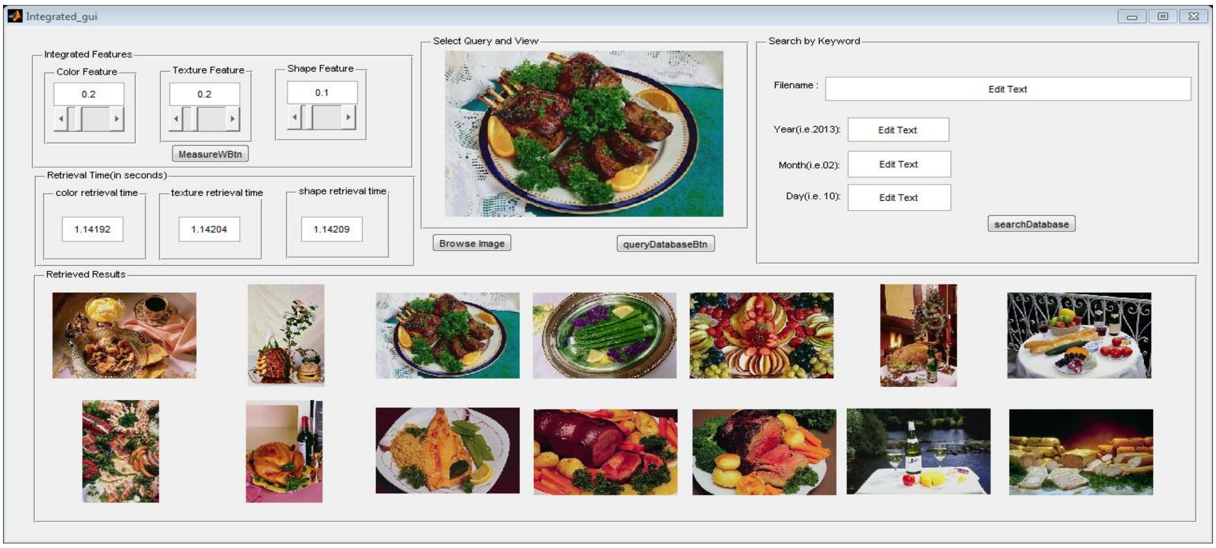


Figure 6.68: Integrated proposed system evaluation: Food class

Algorithm Evaluation: Efficiency and Precision

Figure 6.69 depicts the retrieval time of the four algorithms that were evaluated. The retrieval time for the texture GLCM algorithm is significantly longer compared to the other algorithms, especially in the Bus and Dinosaur classes. The colour histogram also shows significantly longer retrieval times for the Bus and Dinosaur classes, with the integrated algorithm conistently measuring retrieval time of below one second.

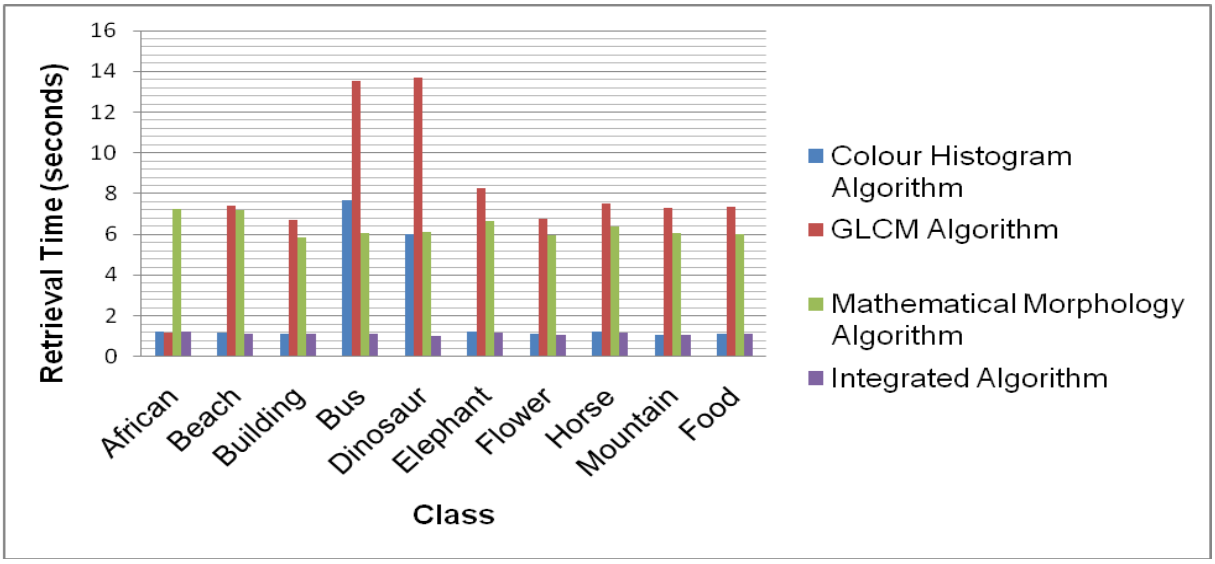


Figure 6.69: Algorithm Efficiency (Retrieval time)

In Figure 6.70 the precision of the algorithms is compared. All of the algorithms except the colour histogram algorithm, that performed poorly in the dinosaur class, proved to be precise. The problem with the colour histogram in the dinosaur class is due to the confusion of colour quantities contained in the image. In order to eliminate this problem, the colour histogram can be improved by combining it with other techniques such as colour moments, colour correlogram and colour coherence vector.

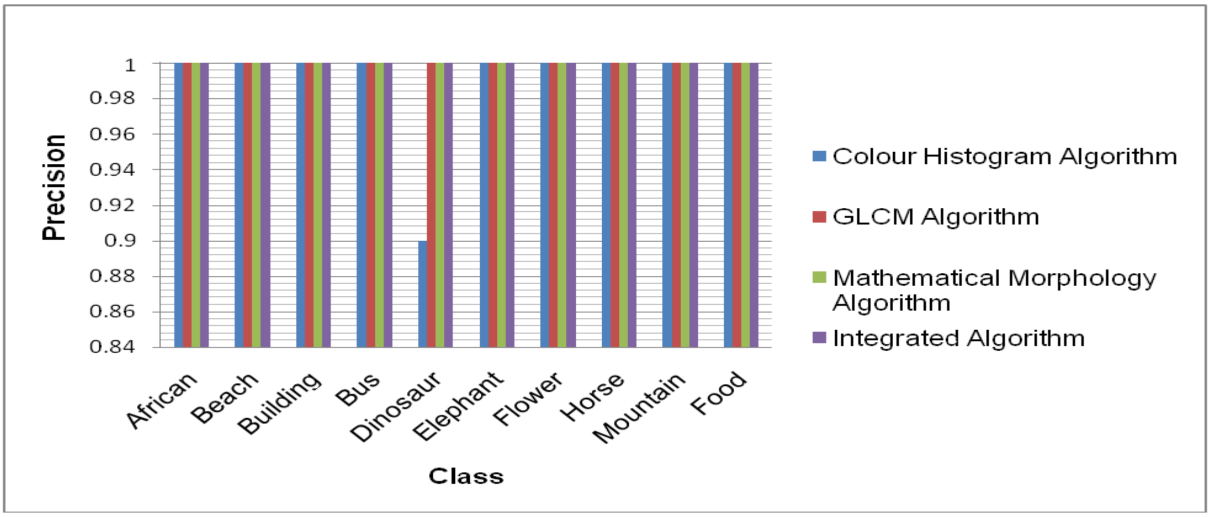


Figure 6.70: Precision of algorithms of the proposed system

E. Comparison of the proposed system with other systems

Apart from evaluating the proposed system against different image classes contained within the Wang database, it is also necessary to evaluate against systems currently available in literature and practice. Implemented systems that also use the Wang image database for a small set of images were selected and the results are listed in Table 6.5.

Table 6.5 shows the validity of integrating several algorithms in order to improve the precision of image classification and retrieval systems. Proposed systems consistently improved as research projects focused on image retrieval problems, especially within the nano-satellite sphere. The performance of the proposed system within the set boundaries and constraints showed the ability to retrieve accurate results with a high precision. Classification of the images to specific groups or classes also improved with the proposed system.

Table 6.5: Comparison: Precision

Class	Lin *et al.* (2009)	Raghupathi *et al.* (2010)	Afifi (2011)	Mikharq (2013)	Proposed System
African	0.68	0.75	0.71	1	1
Beaches	0.62	0.6	0.856	0.93	1
Buildings	0.71	0.43	0.8341	0.61	1
Buses	0.92	0.69	0.8571	1	1
Dinosaurs	0.97	1	0.9975	1	1
Elephants	0.86	0.72	0.7143	0.63	1
Flowers	0.76	0.93	0.9374	1	1
Horses	0.87	0.91	0.5714	1	1
Mountains	0.49	0.36	0.4286	0.65	1
Foods	0.77	0.65	0.9752	1	1
Total	0.762	0.704	0.9752	0.882	1

Figure 6.71 reflects on the comparison of the proposed system against existing systems. The proposed system outperforms other systems in most of the classes.

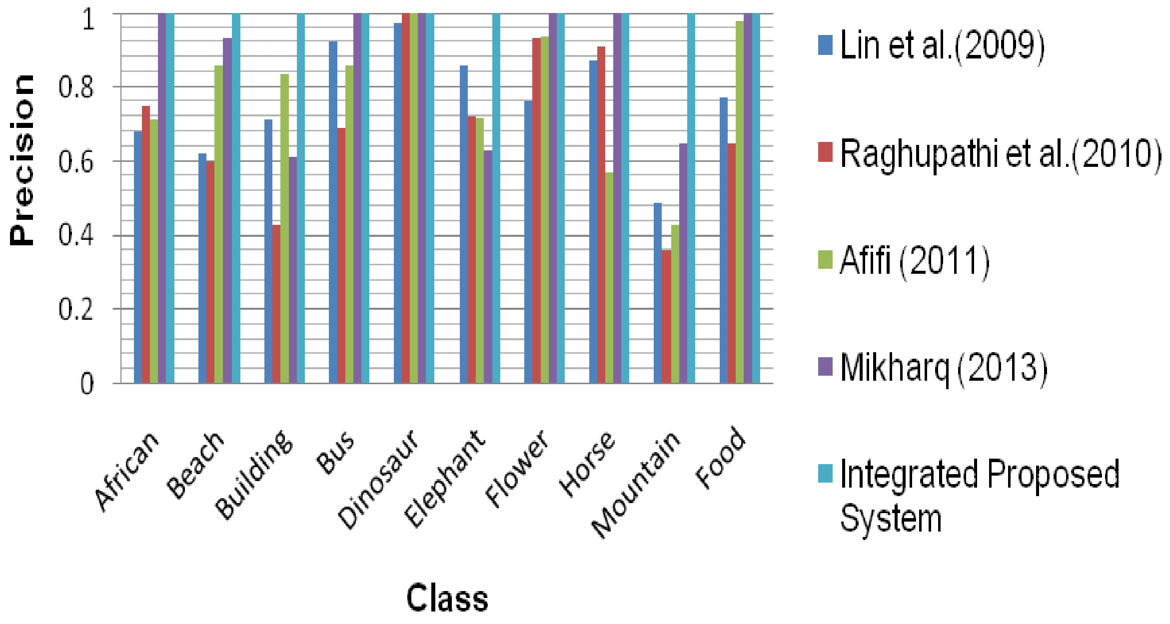


Figure 6.71: Comparison: Precision

F. Evaluation: Additional set groups of the Wang database

Although exceptional results were obtained on small image sets, the proposed system needs to be evaluated using larger groups. Groups of 25, 50 and 100 images (see Figure 6.72) were therefore selected from the Wang database and efficiency and precision measured. These groups were retrieved from the database of 1000 images, that contains ten groups, 100 images per group, to verify the retrieval rate and accuracy of the system. At the start, each group was reduced to 25, 50 and maintained at 100 images per group.

It has to be noted that evaluation results of similar systems using large sets of images are not available. Thus the results obtained for sets of 25, 50 and100 images can only be evaluated in terms of future CubeSat implementations and will serve as basis for future developments.

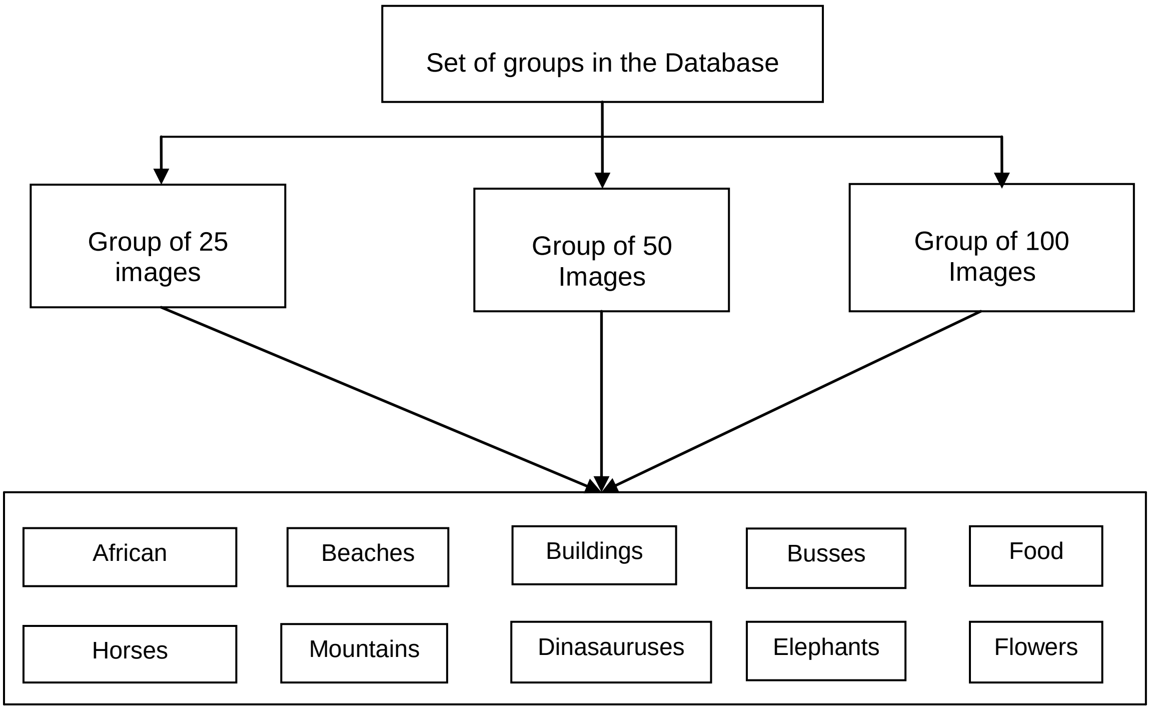


Figure 6.72: Block Diagram groups of images

Figure 6.73 shows the group of twenty five images from the African class that were selected, extracted and stored in the database. All retrieved 25 images corresponds to the features that were specified, resulting in a precision of 1.

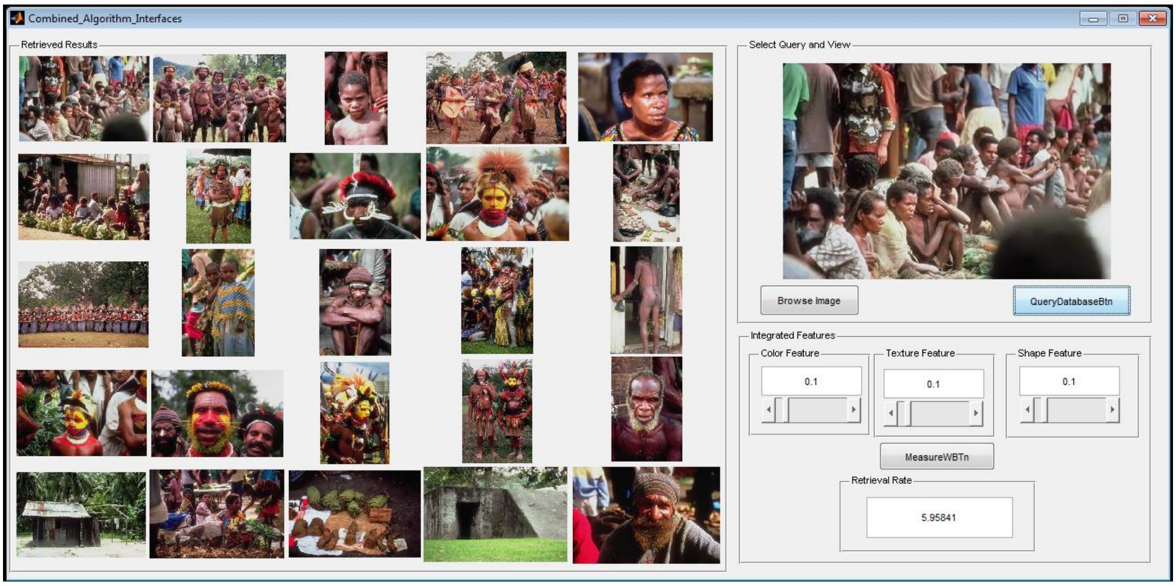


Figure 6.73: African class: 25 images

In Figure 6.74 fifty images from the African class are selected, extracted and stored in the database. From the fifty retrieved results, 42 images are relevant to African class and eight images from the Beaches class.

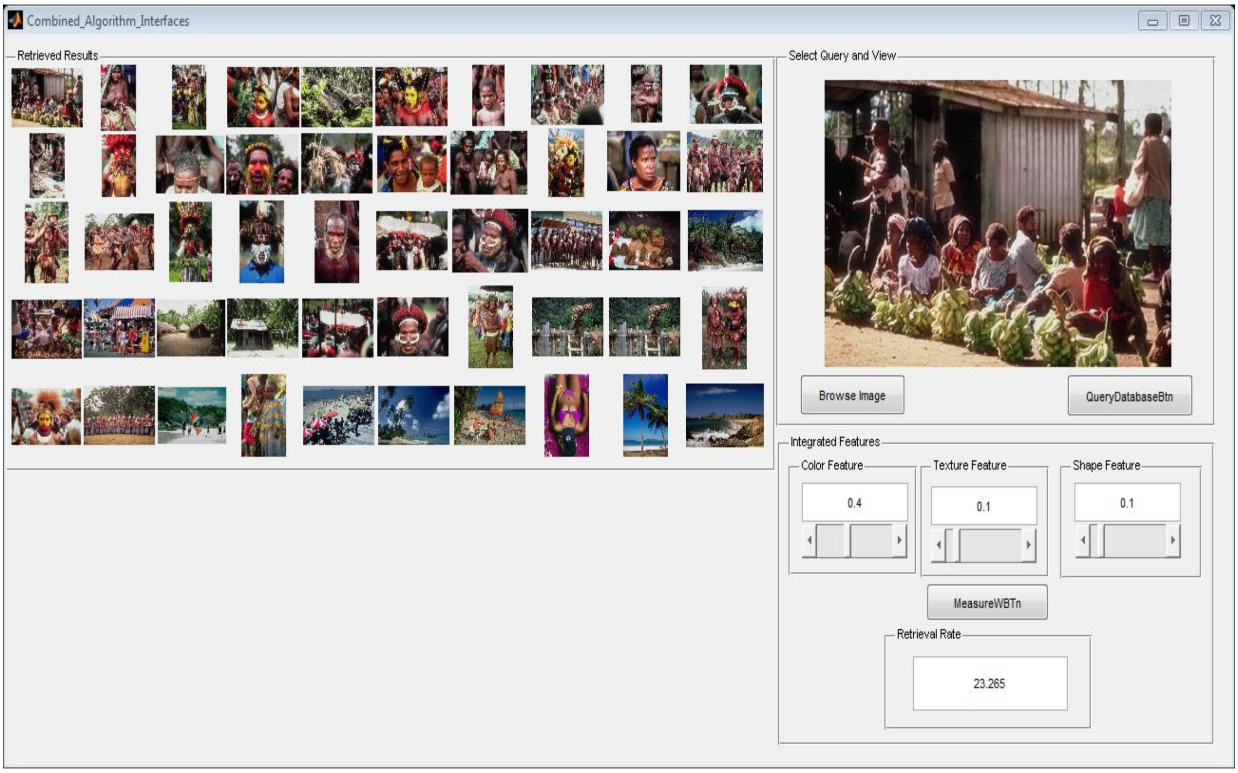


Figure 6.74: African class: 50 images

Figure 6.75 shows the one hundred images selected, extracted and stored in the database. A single image (query) was selected to help find similar images in the database. Out of one hundred images retrieved, 98 images are relevant to the African class and two images belong to the Beaches class.

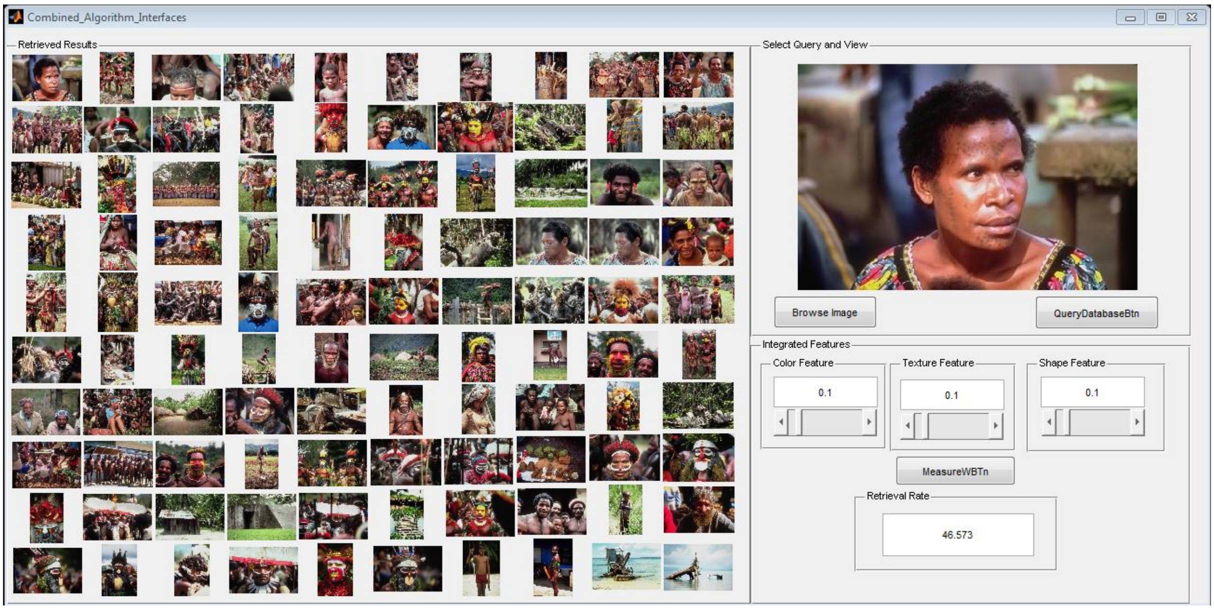


Figure 6.75: African class: 100 images

In Figure 6.76, a query image was selected from the Beaches class of twenty five images and submitted to the system. The retrieved results were twenty five of which eighteen are relevant to the queried image and seven are irrelevant. The precision is 0.72 (18/25).

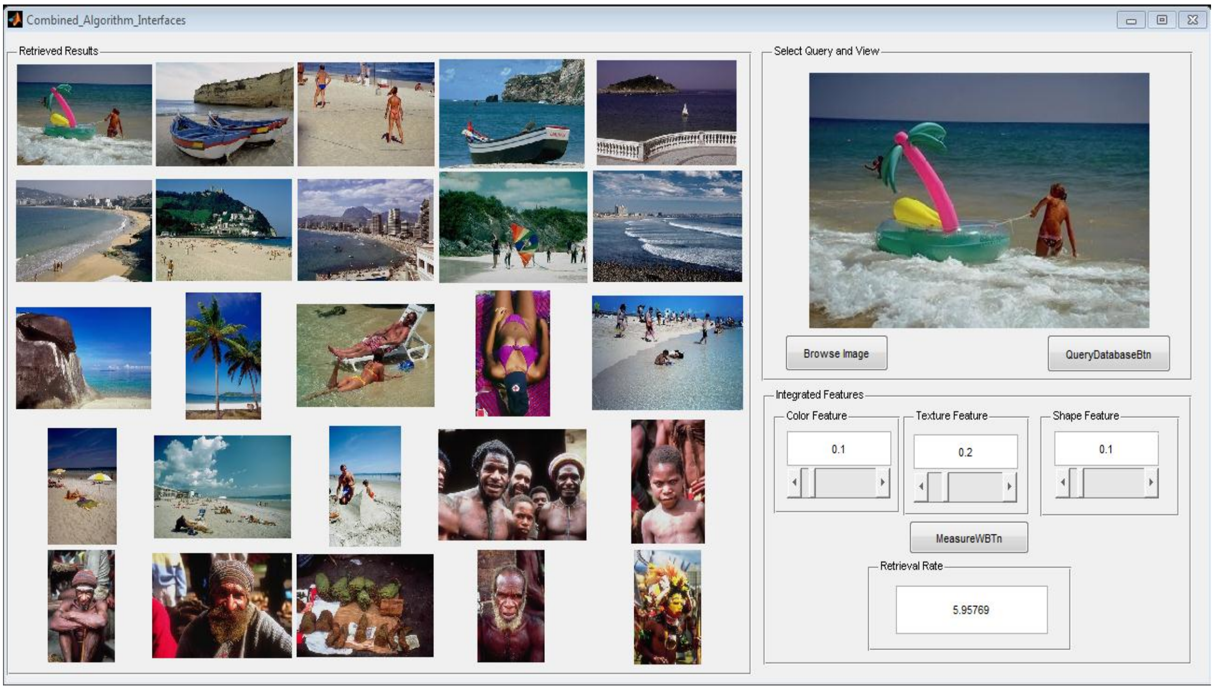


Figure 6.76: Beaches class: 25 images

Figure 6.77 shows a group of fifty beach images as well as the query image. The obtained results are 32 relevant and eighteen images that are not relevant. The precision is 0.64 (32/50).

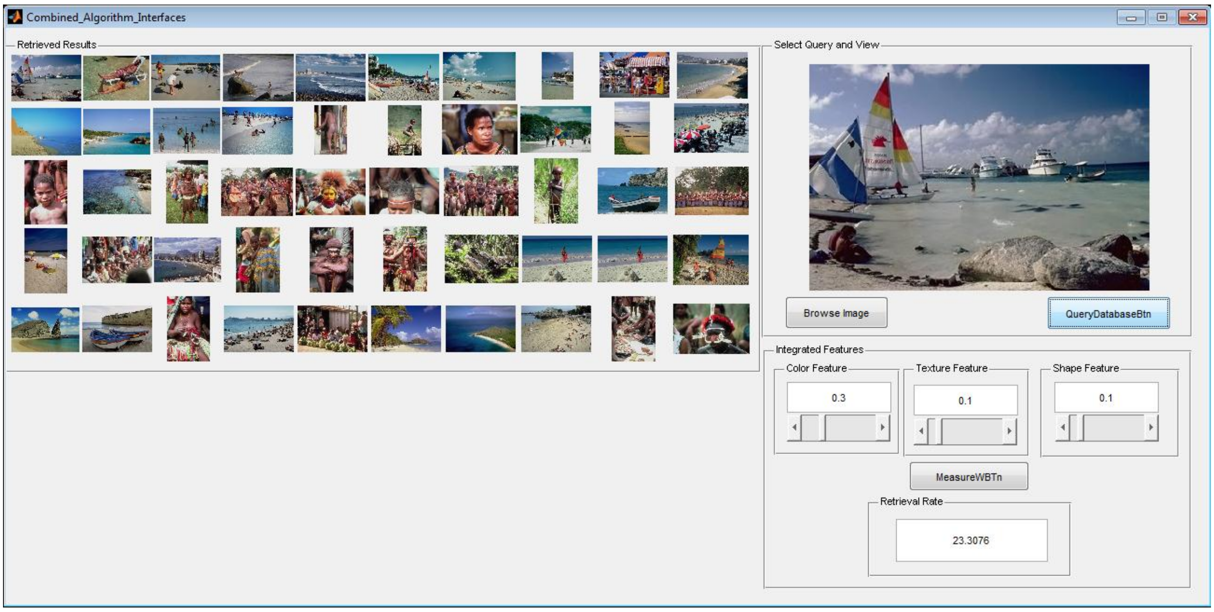


Figure 6.77: Beaches class: 50 images

Figure 6.78 shows a group of 100 beach images. with the obtained results of 58 relevant to the queried image and 42 irrelevant. Precision is 0.58 (58/100).

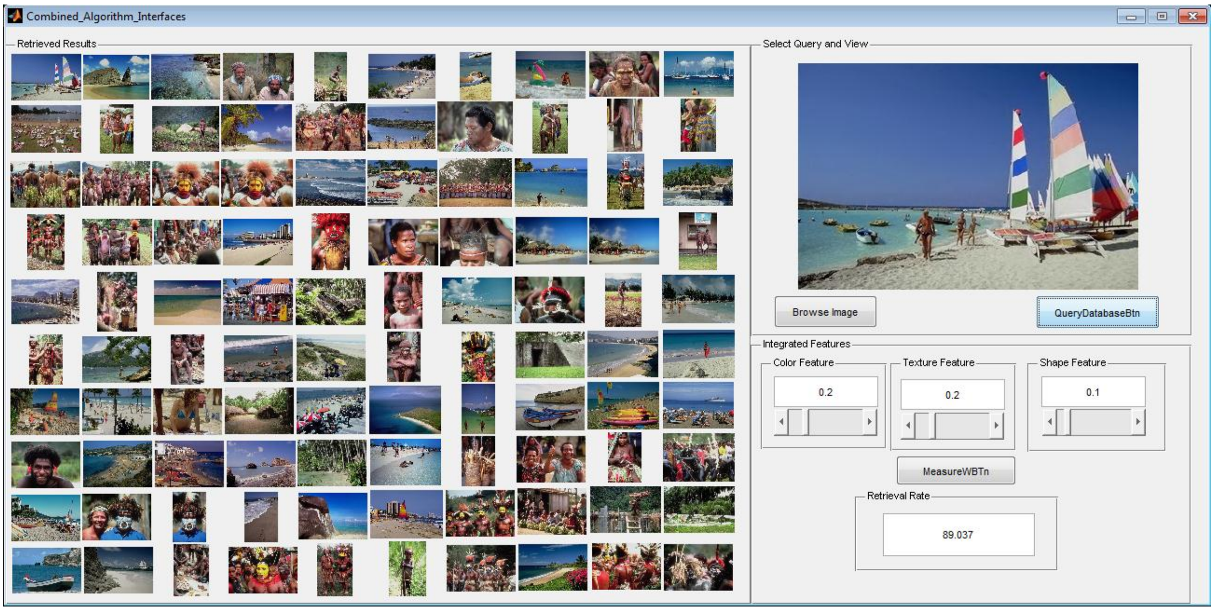


Figure 6.78: Beaches class: 100 images

Figure 6.79 shows a group of 25 building images. The retrieved results are all relevant to the query and therefore precision is 1 or 100%.

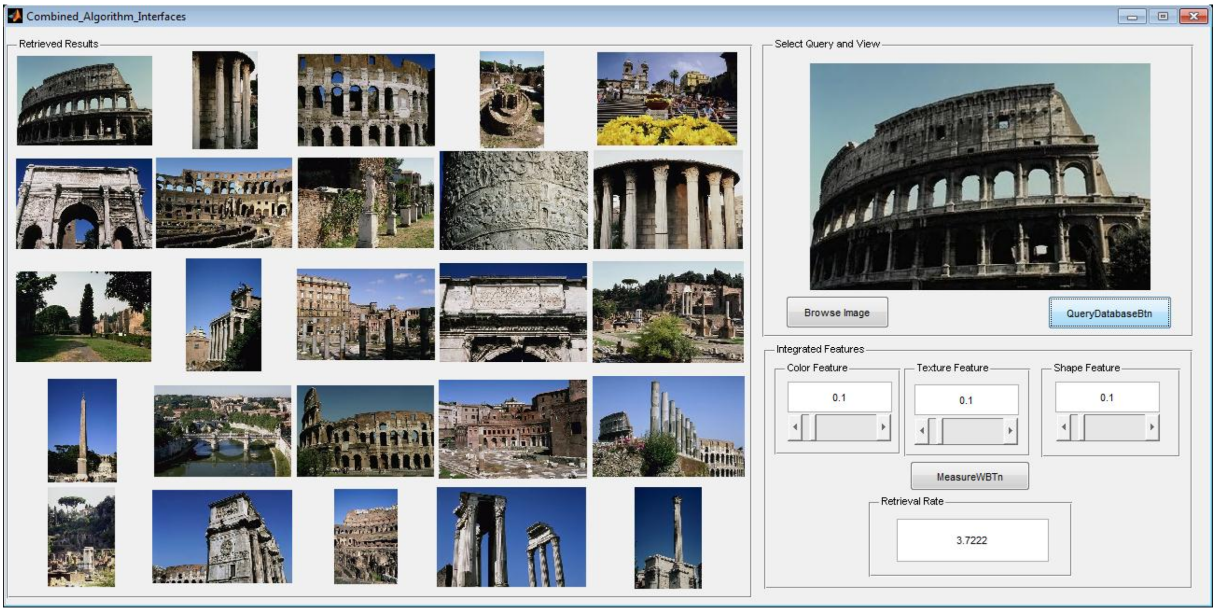


Figure 6.79: Building class: 25 images

In the building class, 50 images were extracted and stored in the database. Figure 6.80 shows the single building image used as the selected query within the view panel. Retrieved results shows precision of 0.44 (22/50).

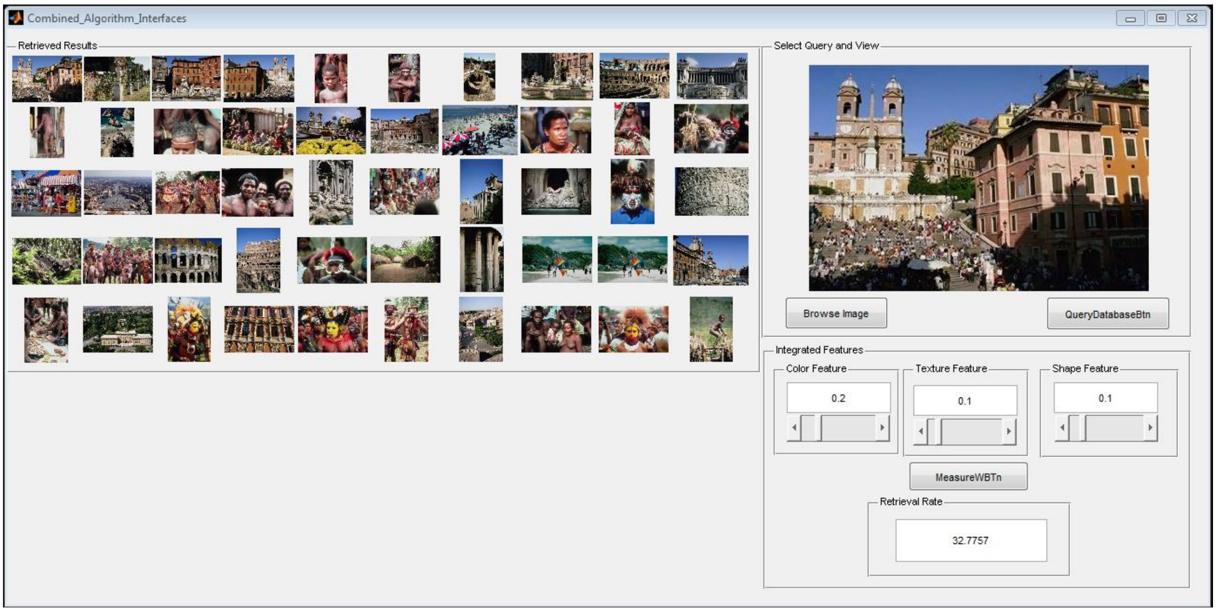


Figure 6.80: Building class: 50 images

Figure 6.81 shows the building class containing 100 images, that were extracted and stored in the database. A single building image was used as query to find similar images in the database. Of the retrieved results, 31 were relevant to the query and 69 were irrelevant. The precision is 0.32 (32/100).

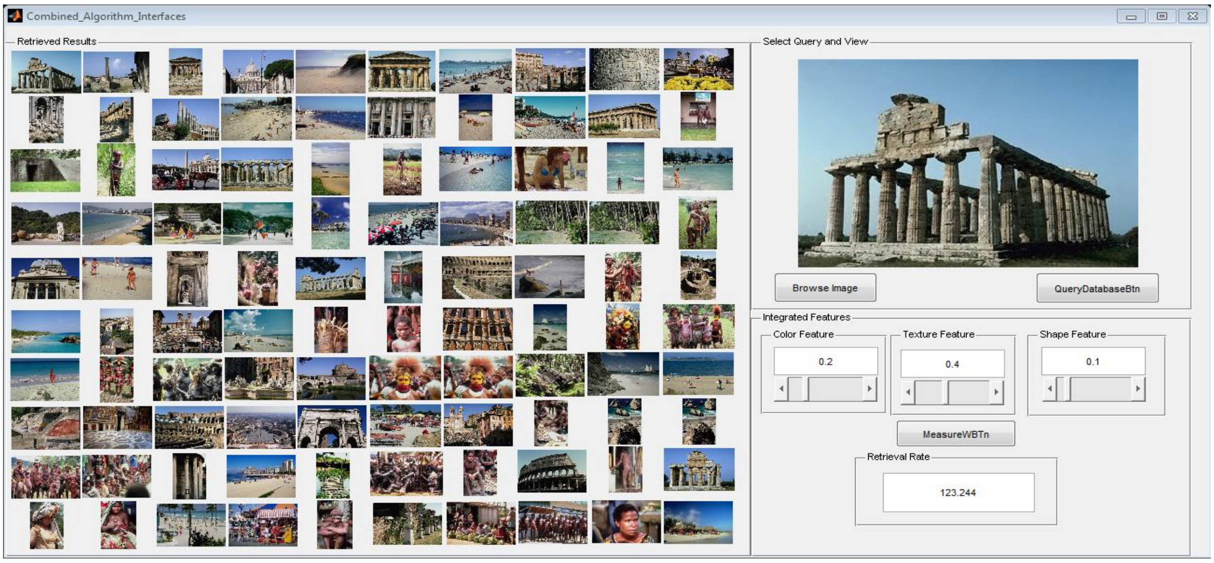


Figure 6.81: Building class: 100 images

Figure 6.82 reflects on a retrieved precision of 1 for 25 images from the Busses class. The same results were obtained for 50 images (see Figure 6.83).

Figure 6.82: Busses class: 25 images

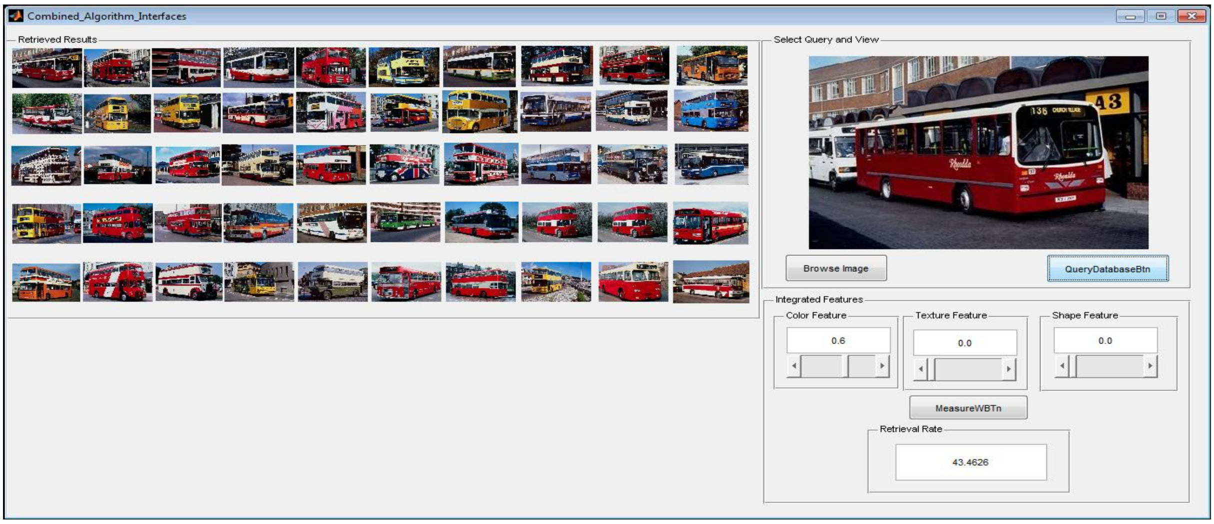


Figure 6.83: Busses class: 50 images

Figure 6.84 shows the precision of 37 relevant images from a possible 100.

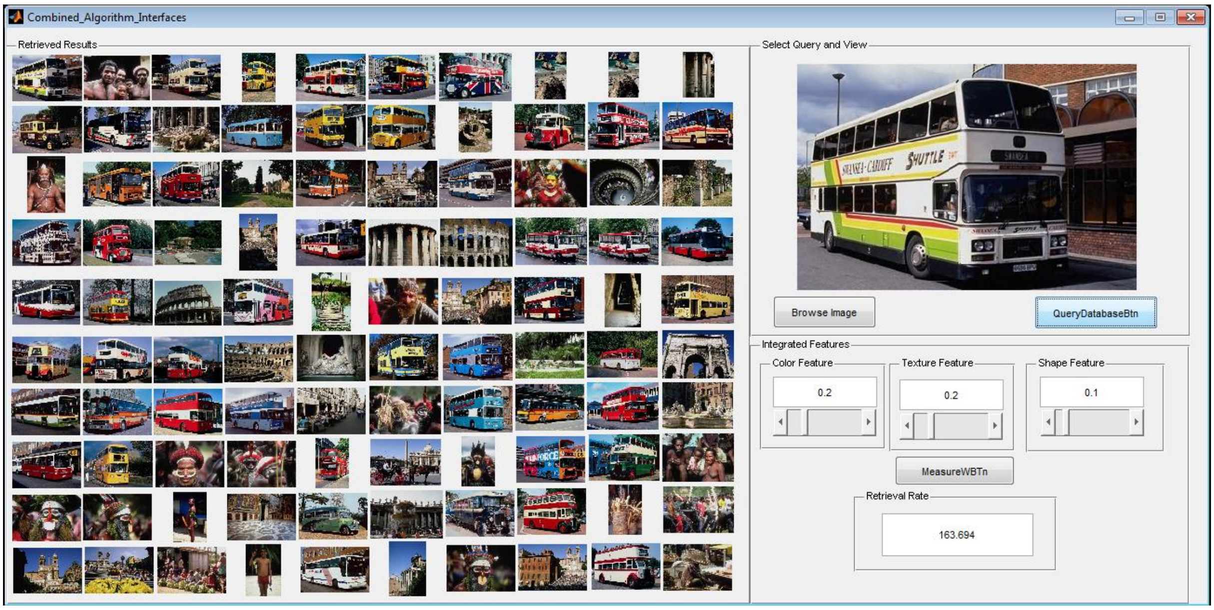


Figure 6.84: Busses class: 100 images

Within the Dinosaurus class, precision of 1 were obtained for 25 images (see Figure 6.85) and 50 images (Figure 6.86).

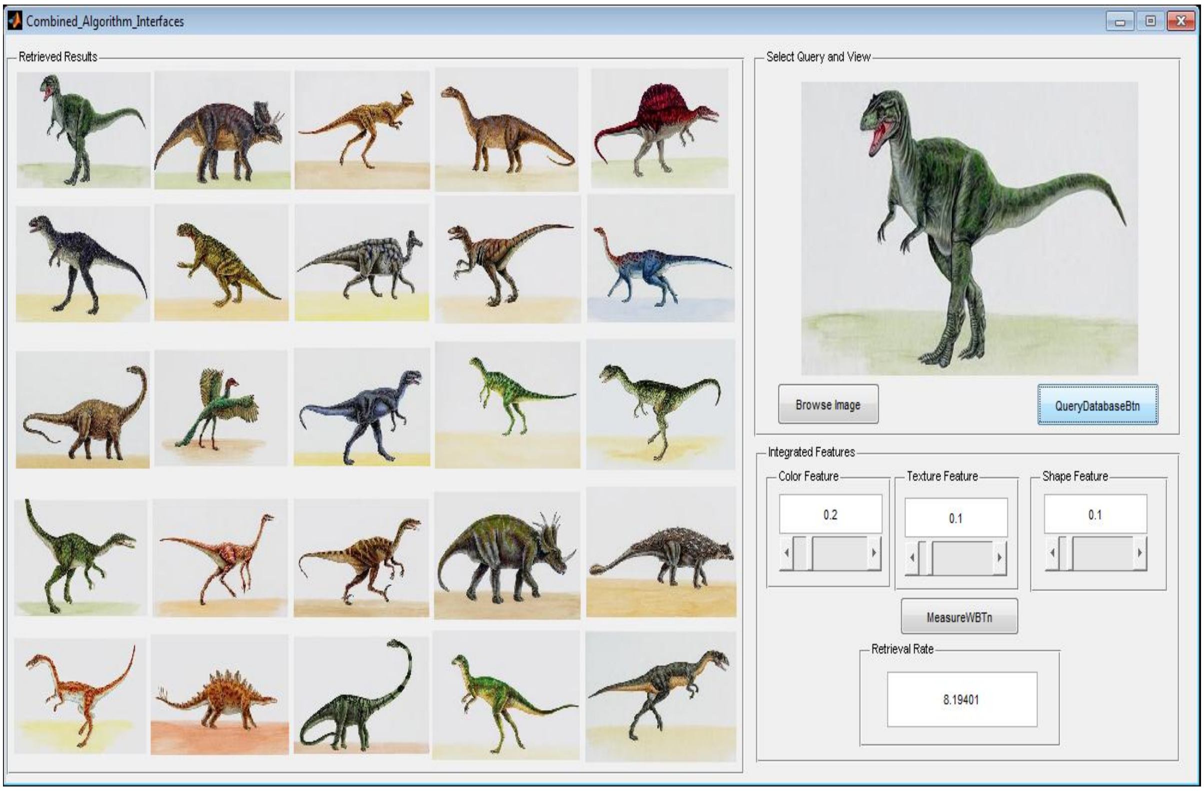


Figure 6.85: Dinosaur class: 25 images

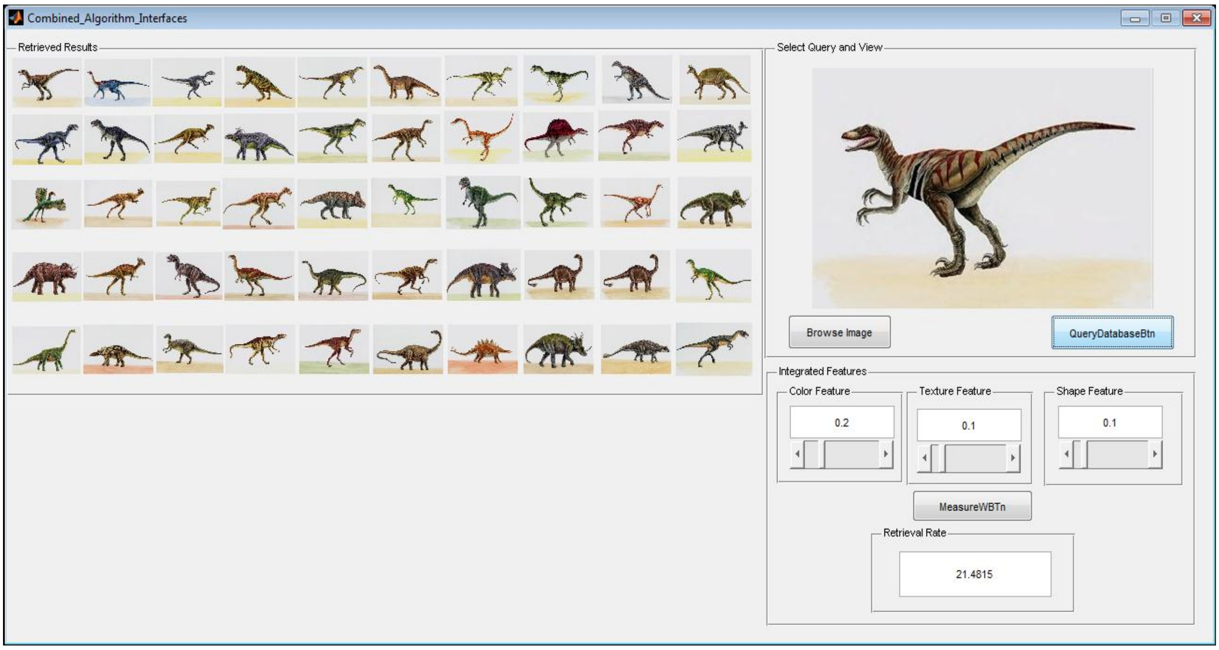


Figure 6.86: Dinosaur class: 50 images

Within the group of 100 images, precision of 0.98 was recorded (see Figure 6.87).

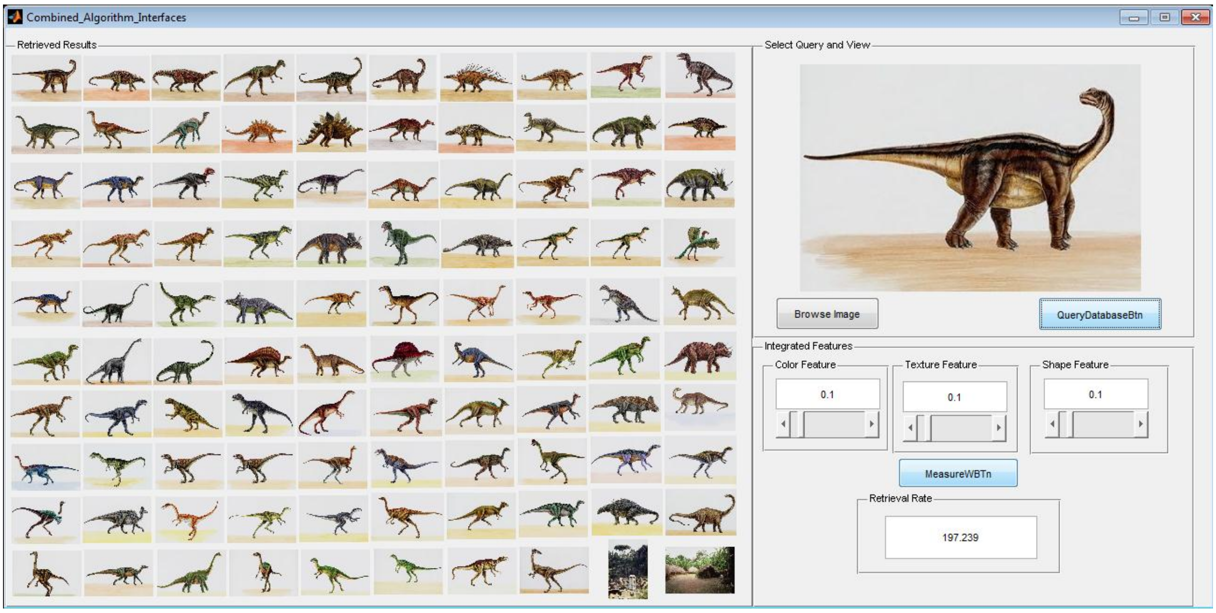


Figure 6.87: Dinosaur class: 100 images

Figure 6.88 shows a precision of 0.8 for 25 images of the Elephant class, 0.76 for 50 images (see Figure 6.89) and 0.42 for 100 images.

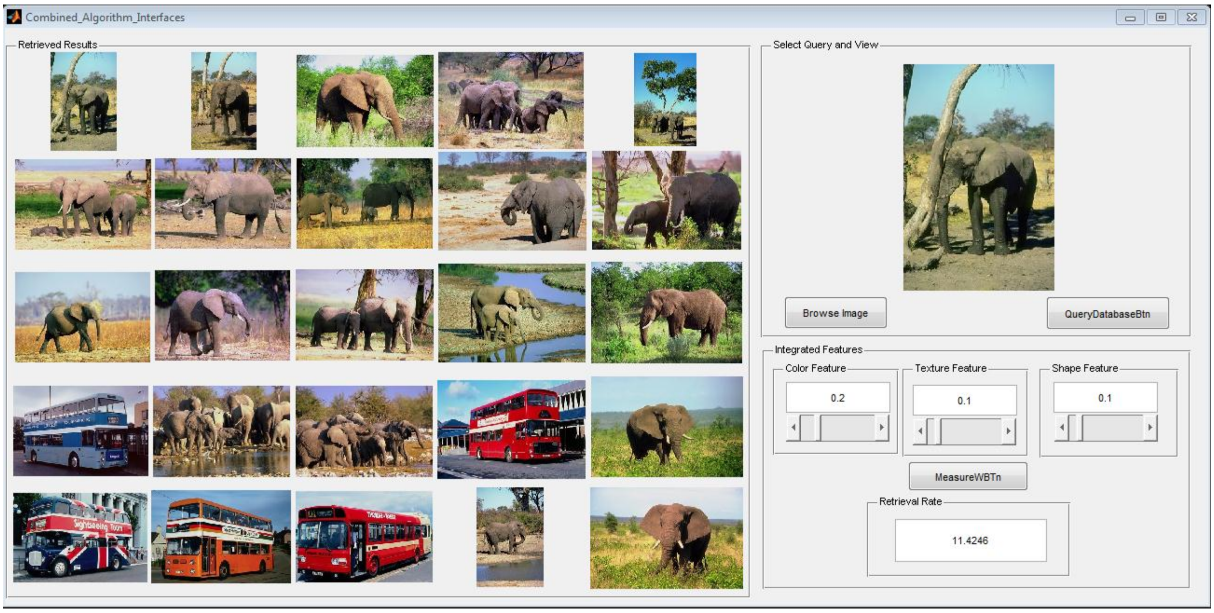


Figure 6.88: Elephant class: 25 images

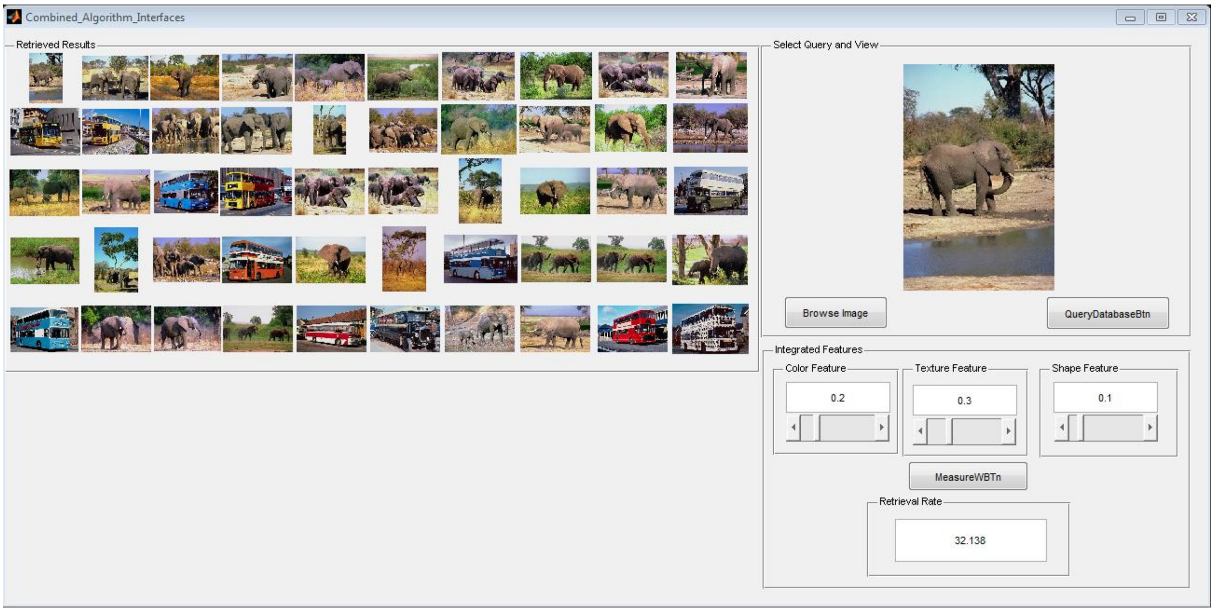


Figure 6.89: Elephant class: 50 images

Figure 6.90 shows a group of Elephant images, which contains 100 images, that were extracted and stored. A query image is used to query the database. Out of one hundred images retrieved, 42 images are relevant to the query and 52 images are irrelevant to the query. Therefore, precision is 0.42 or 42%.

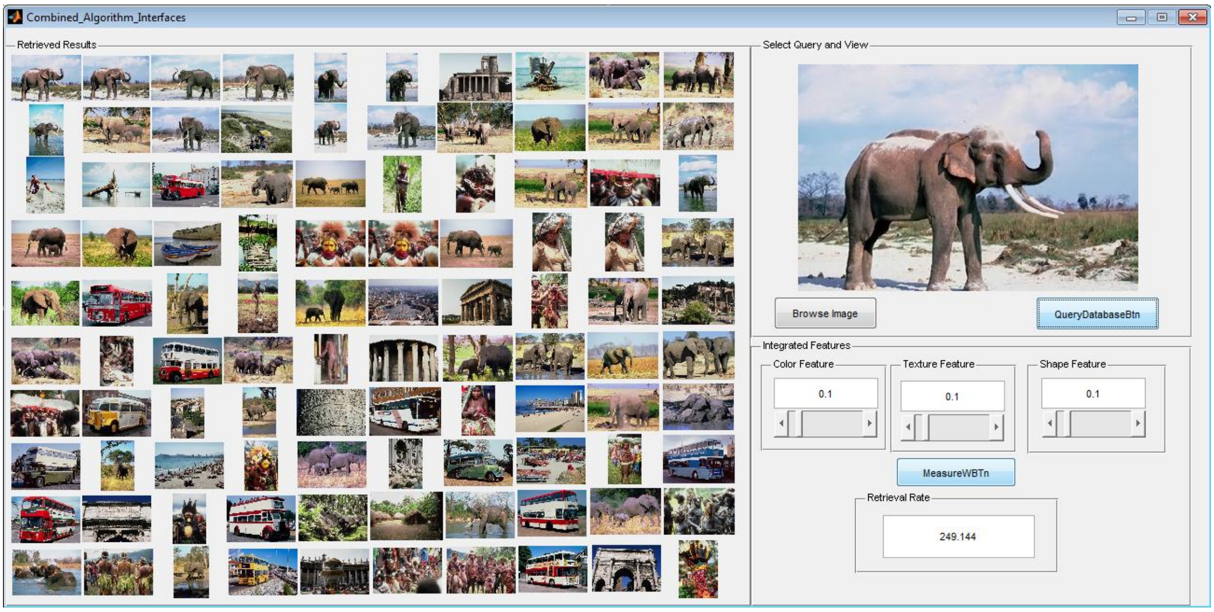


Figure 6.90: Elephant class: 100 images

The flower class recorded precision values of 0.64 for 25 images (Figure 6.91), 0.6 for 50 images (Figure 6.92) and 0.55 for 100 images (Figure 6.93).

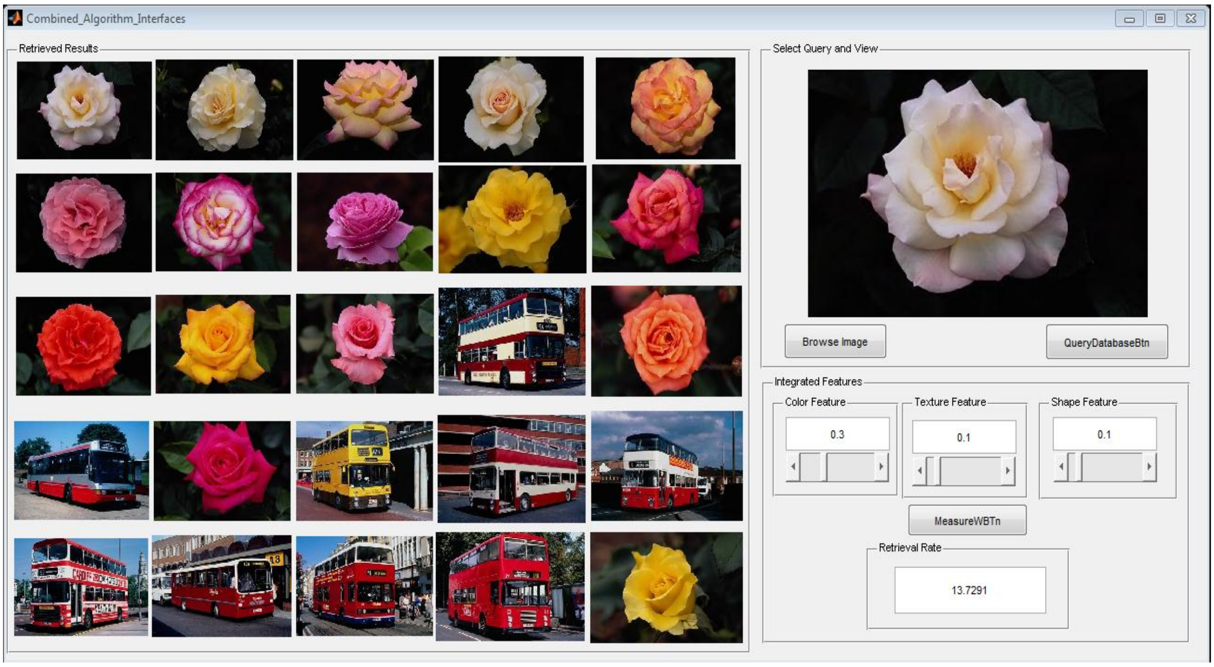


Figure 6.91: Flower class: 25 images

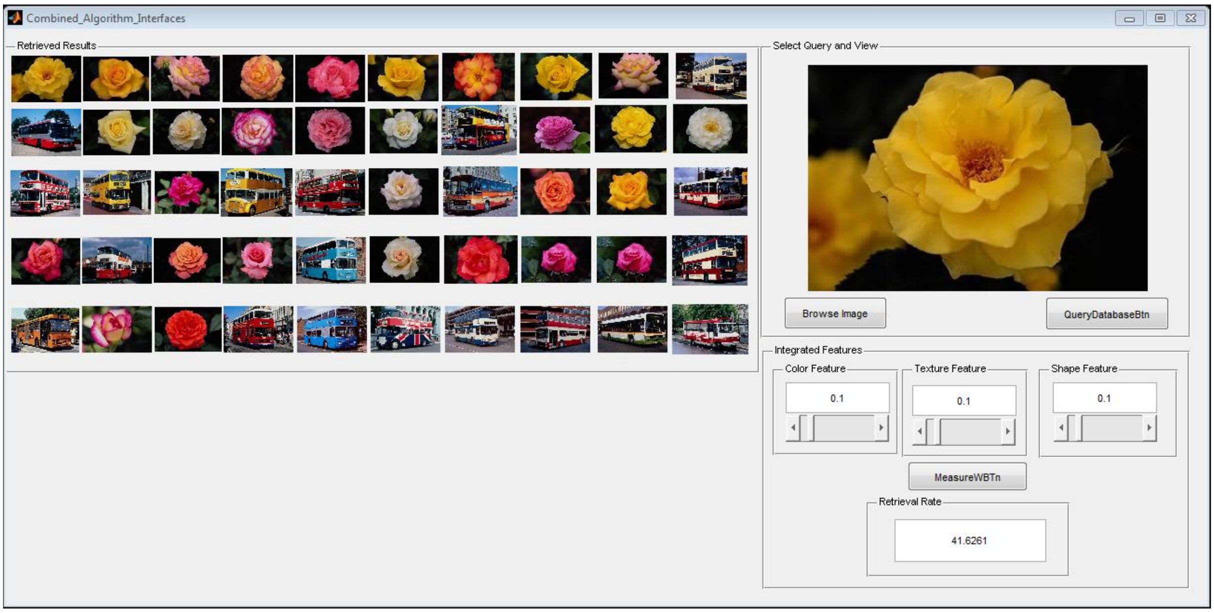


Figure 6.92: Flower class: 50 images

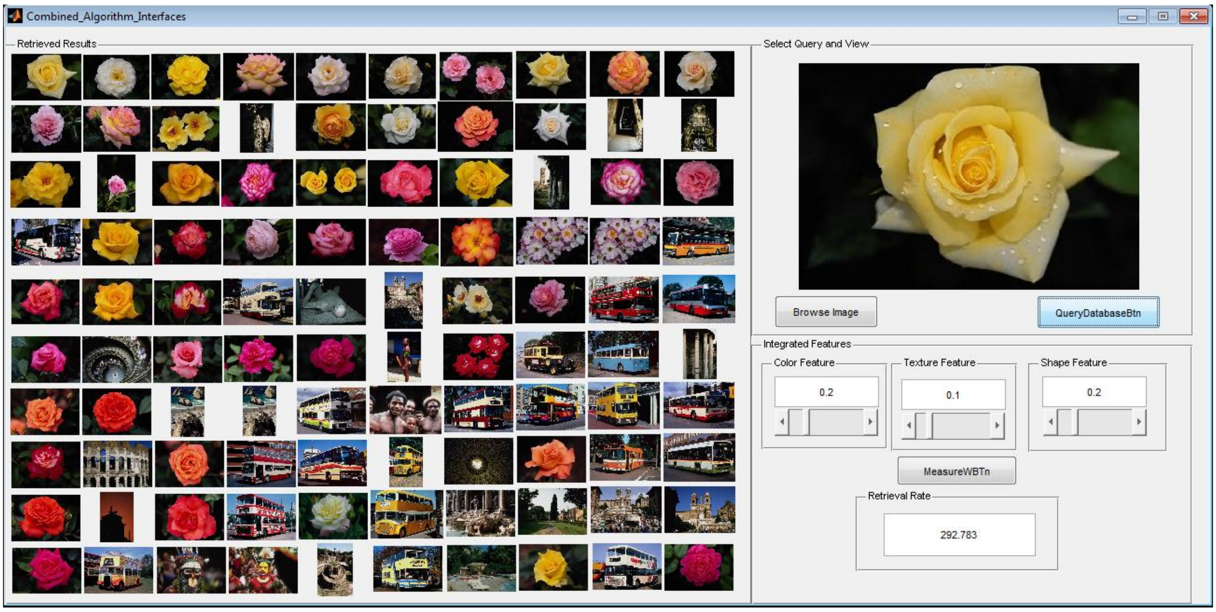


Figure 6.93: Flower class: 100 images

The Horses class shown in Figure 6.94, Figure 6.95 and Figure 6.96 recorded precisions of 0.2 for 25 images, 0.24 for 50 images and 0.42 for 100 images.

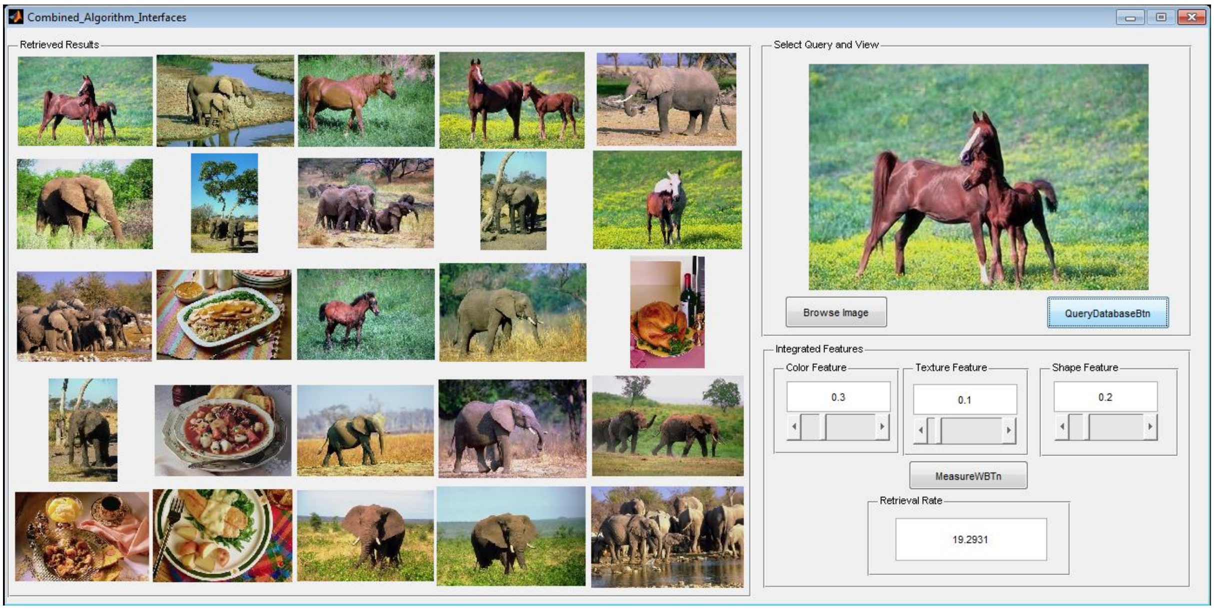


Figure 6.94: Horse class: 25 images

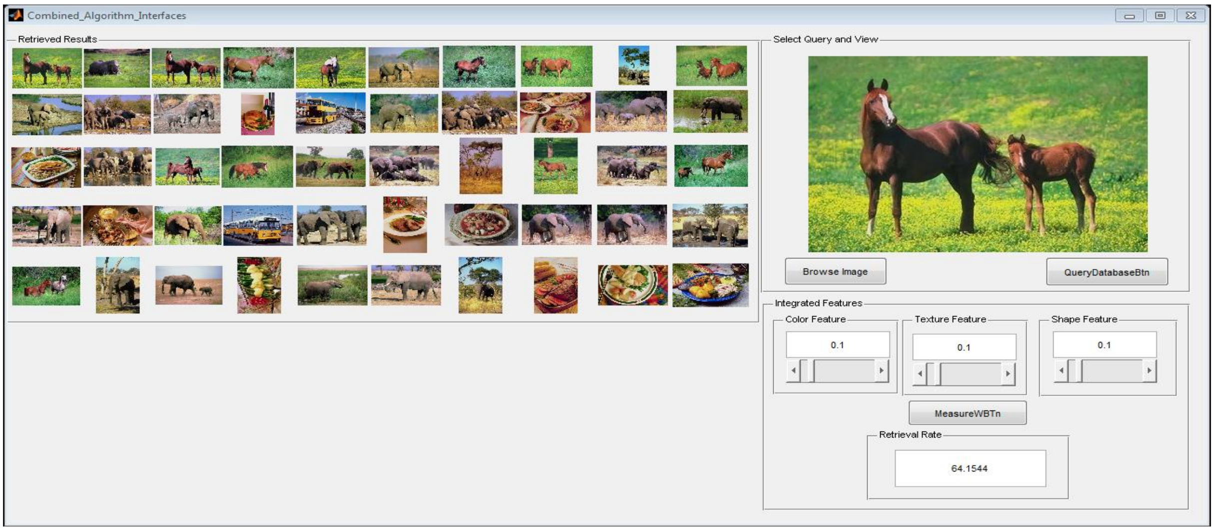


Figure 6.95: Horse class: 50 images

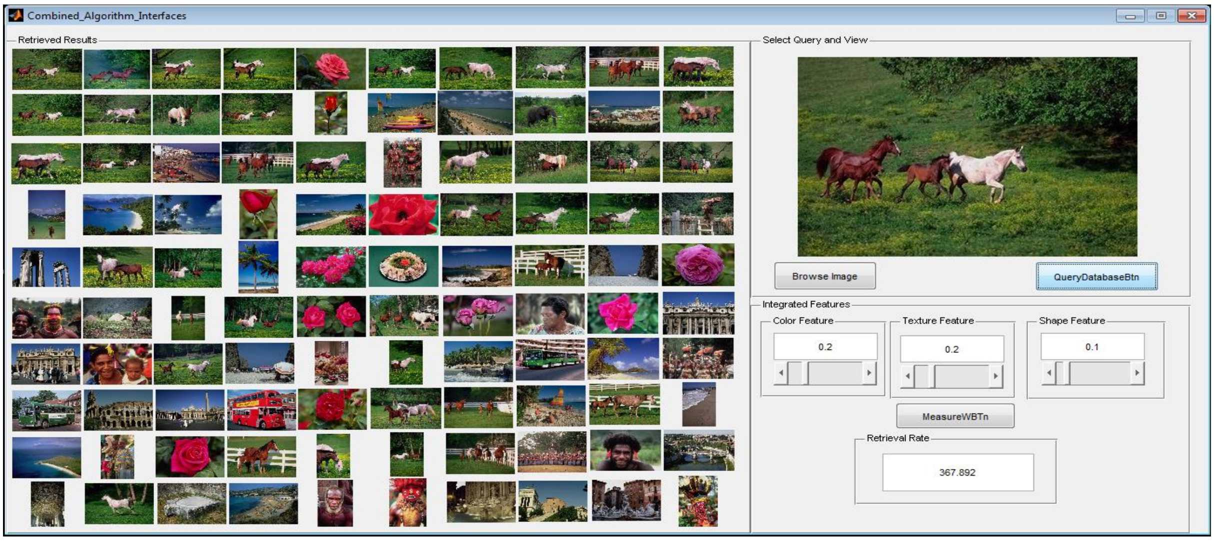


Figure 6.96: Horse class: 100 images

Figure 6.97 shows a group of twenty five images from the Mountain class. From the twenty five images, seven images are relevant to the query and eighteen images are irrelevant to the query. Therefore, precision is 0.28 (7/25).

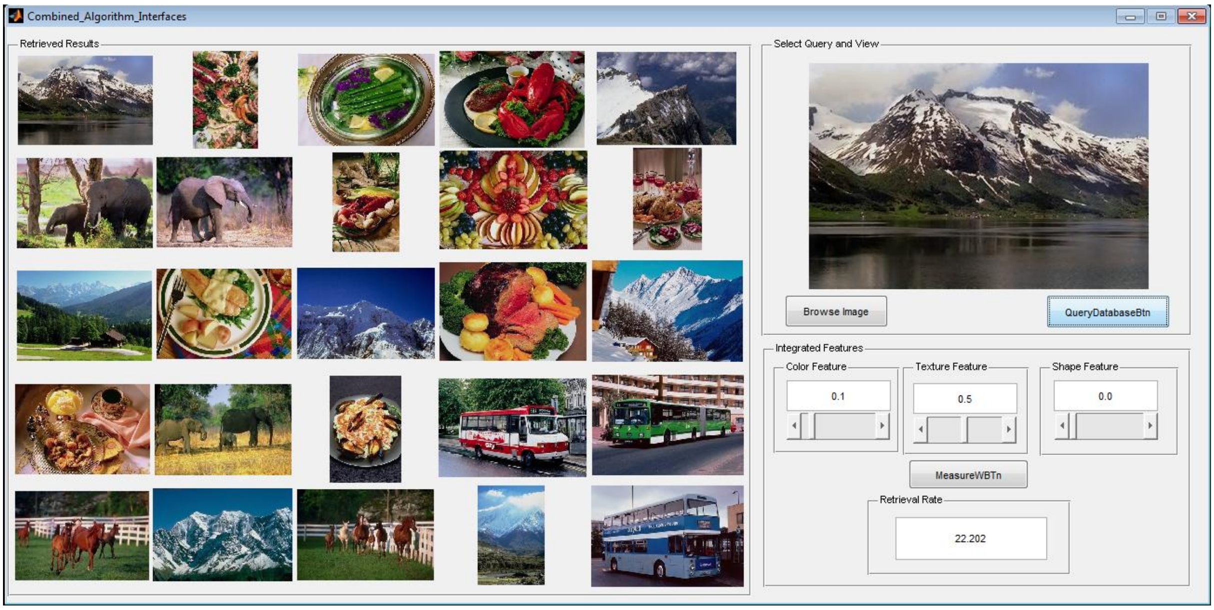


Figure 6.97: Mountain class: 25 images

Figure 6.98 shows the 50 images from the Mountains class with a recorded precision of 0.28.

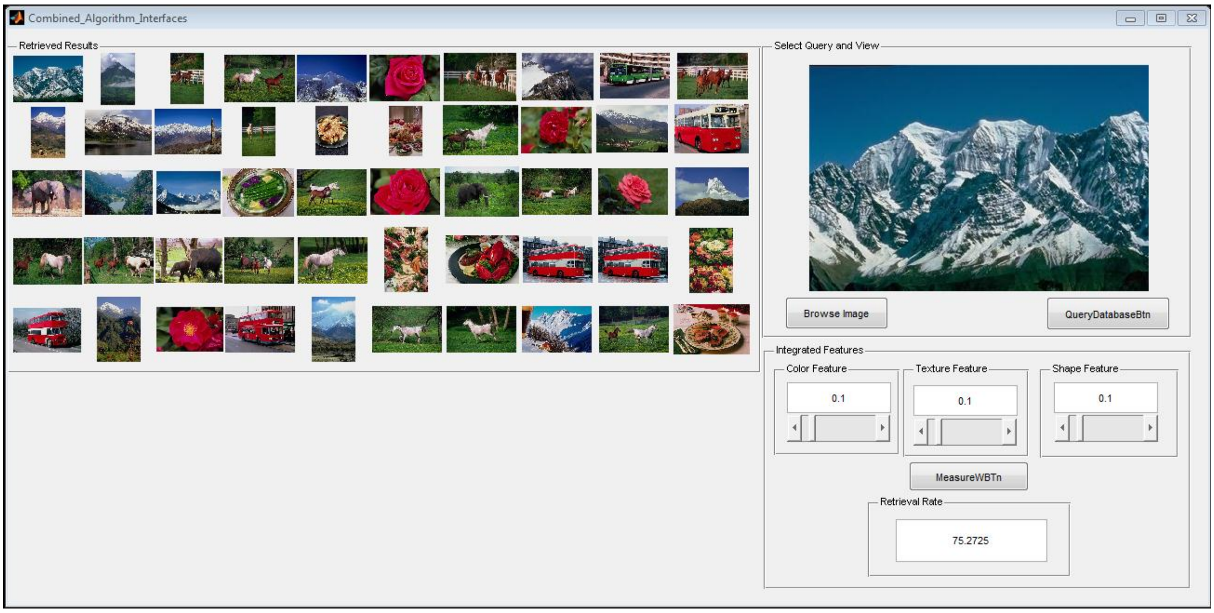


Figure 6.98: Mountain class: 50 images

Figure 6.99 indicate a precision of 0.09 for 100 images.

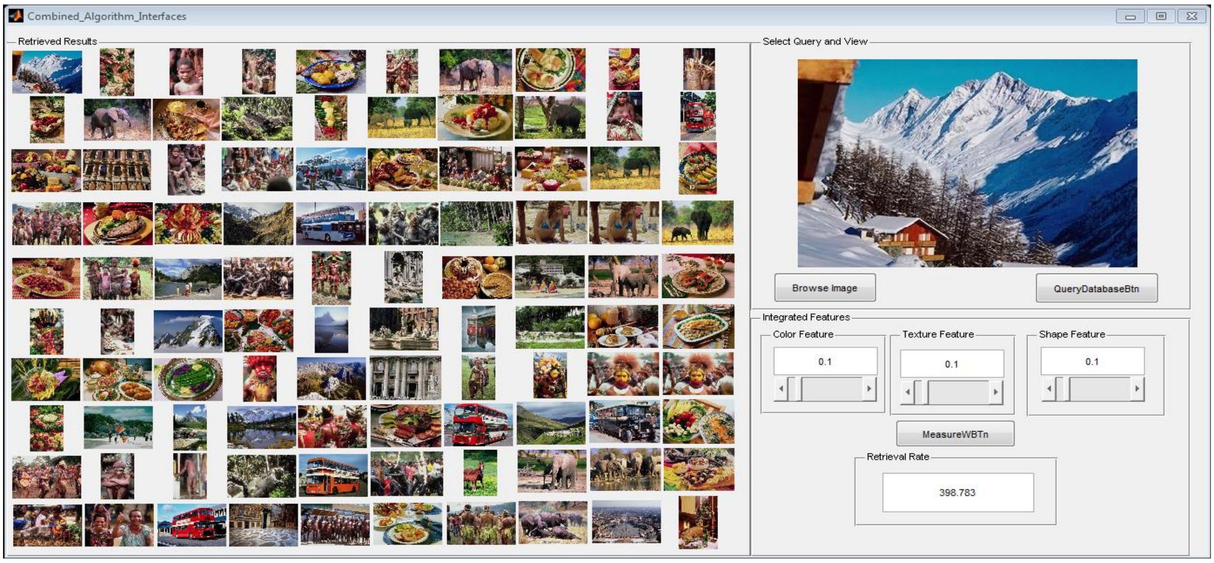


Figure 6.99: Mountain class: 100 images

Within the Food class, precision results of 0.56 for 25 images (Figure 6.100), 0.5 for 50 images (Figure 6.101) and 0.26 for 100 images (Figure 6.102) were recorded.

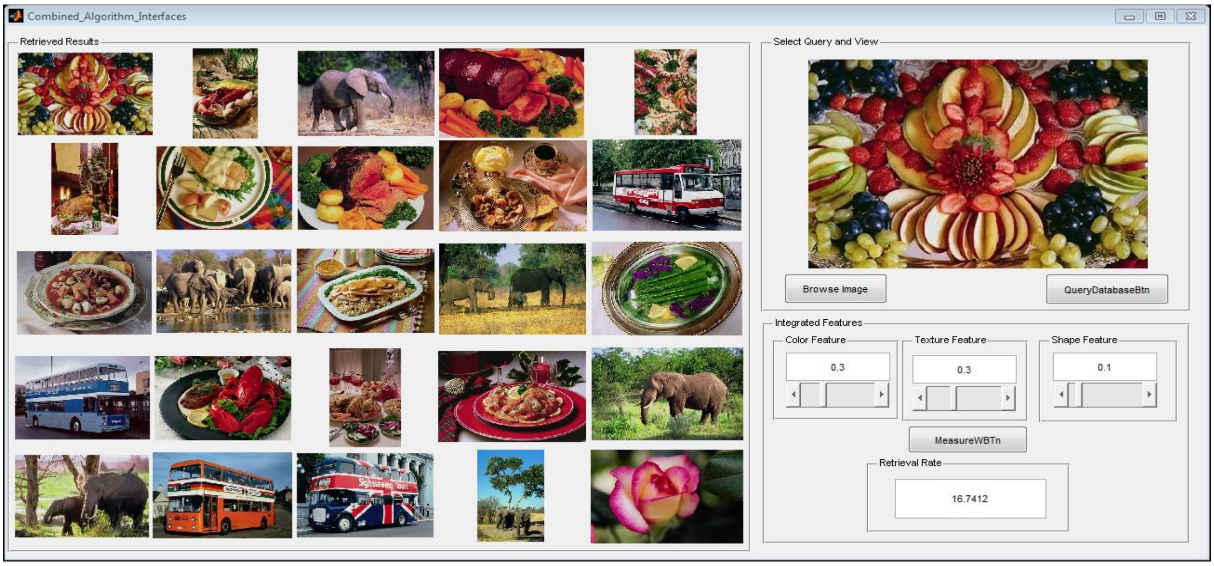


Figure 6.100: Food class: 25 images

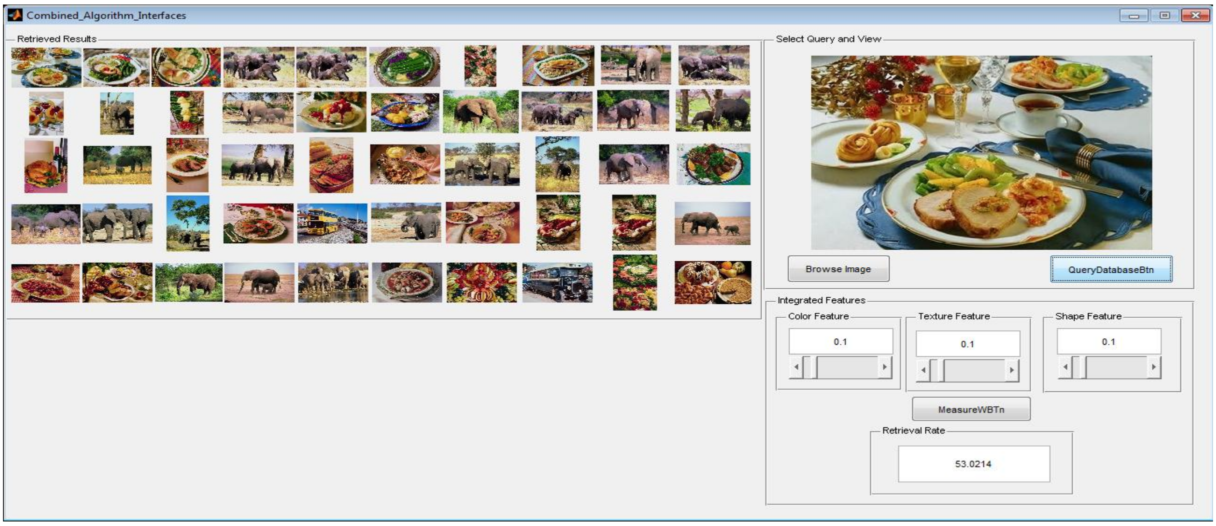


Figure 6.101: Food class: 50 images

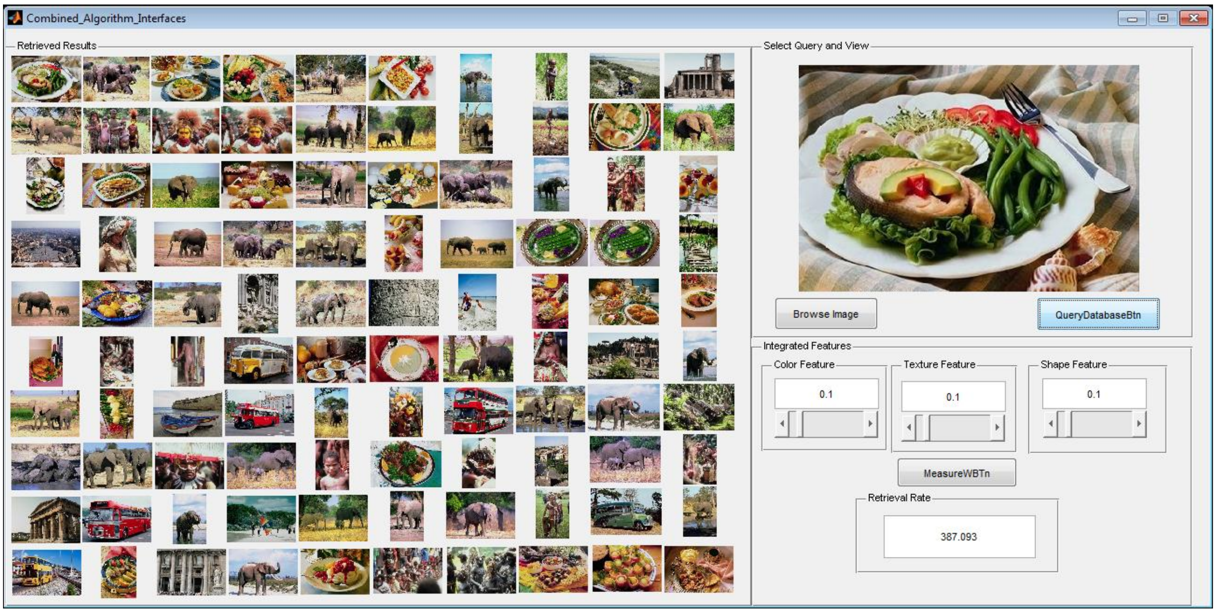


Figure 6.102: Food class: 100 images

Table 6.6 put forward evaluation results for the different groups of images (sets of 25, 50 and 100) within the Wang Database of 1000 images. These sets were used to verify accuracy and retrieval time for the system, within different groups. Each set contains ten classes, meaning a set of 25 relates to 250 images.

When selecting a query image to find similar images in the database, always avoid images which tend to give less similarity to the others in the group. In these cases, low precision (of 0.09 in the mountain class) can be the result of bad choices of query image used to find similar images in the database. Usually, corel sets contain images of the same subject, but many groups contain a few images that are visually dissimilar.

Table 6.6: Retrieval time and precision for larger groups

Class	**Retrieval time (25) in seconds**	**Precision (25)**	**Retrieval Time (50) in seconds**	**Precision (50)**	**Retrieval Time (100) in seconds**	**Precision (100)**
African	5.95841	1	23.265	0.84	46.573	0.98
Beaches	5.95769	0.72	23.3076	0.64	89.037	0.58
Buildings	3.7222	1	32.7757	0.44	123.244	0.32
Buses	5.83393	1	42.4626	1	163.694	0.37
Dinosaurs	8.19401	1	21.4815	1	195.239	0.98
Elephants	11.4246	0.80	32.138	0.76	249.144	0.42
Flowers	13.7291	0.64	41.6261	0.60	292.783	0.55
Horses	19.2931	0.20	64.1544	0.24	367.892	0.42
Mountains	22.202	0.28	75.2725	0.28	398.783	0.09
Foods	16.7412	0.56	53.0214	0.50	387.093	0.26
Average Total	11.306	0.72	40.950	0.63	231.35	0.497

Figure 6.103 shows the retrieval time for the different set of groups of 25, 50 and 100 images. As is expected the retrieval time increases as the number of images increases. The retrieval times also differ within the different image classes, as it is dependent on the different colour, shape and texture computations. For instance, the Horse, Mountain and Food classes displayed significantly higher retrieval times than the other classes. This is in part also due to the infrastructure (processing power and memory) limitations used for implementation purposes. For example, upon doubling the memory and internal buffers, the retrieval time decreases by at least 25%. Images to be received from the CubeSat are estimated to be not more than 100 images per initial class, thus putting the estimated retrieval time as per the evaluations in usable context.

Takeda, Kise and Iwamura (2011) stated that a scaled up database, requires a large amount of memory, and the accuracy of retrieval decreases because of lack of discriminating power of the features. To improve such a system, they proposed three methods, namely memory reduction by sampling feature points, improvement of discrimination power by increasing the number of feature dimensions, and stabilizing features by reducing redundancy. They experimented with a database of 10 million image pages and their system achieved 50% in memory reduction, accuracy of 99.4% and took processing time of 38ms.

Krishnamachari and Abdel-Mottaleb (1998) stated that search time increases linearly, depending on the size of the database, when features extracted from a query image are compared to those features stored in a database. One solution to resolve this problem is to use binary representation techniques to speed up the feature comparison. The time taken to compare two images is short, but when a query is compared to all the images in the database, the time taken increases, and an increase in the size of the database will result in long search delays, thus not suitable for practical purposes. It is still an open problem when searching for information in large image databases. Although, common search engines still compare a query against all images in the database, and retrieved results are ranked and sorted. This way doesn't scale up for large databases. To compute the retrieval time for a query against all images in the database, equation 6.1 can be used and it includes time for taken for sorting and ranking all the retrieved results.

$$T_{total} = nT_{1sim} + O(n\log n) \qquad \text{Equation 6.1}$$

where

n is the number of images stored

T_{sim} is the time takes for similarity of two images

$O(n\log n)$ s sorting time for n elements.

T_{sort} is the time takes to rank images

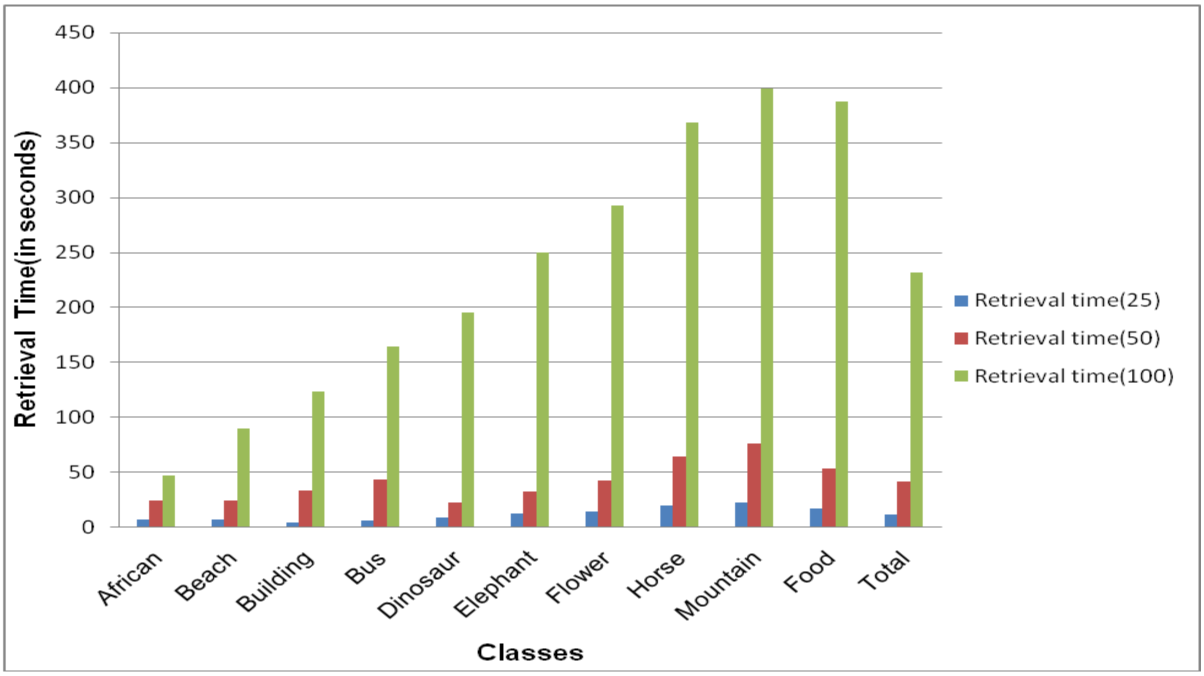


Figure 6.103: Comparison of set group retrieval time

Figure 6.104 shows a comparison of the precision of the sets of groups of images at 25, 50 and 100 images per group. The set of the group of 25 images produced higher results with an average precision of 0.72, compared to 0.63 within the group of 50 images and 0.497 within the group of 100 images. Specific classes, such as the Horse and Mountain class, produced below average results. This is, however, in line with research results indicating significantly lower precision in Horse and Mountain classes (Singha & Hemachandran, 2012). This is classified as a high level classification problem and proposals to counter this include measuring the saliency of features based on the plot of the inter-class and intra-class distributions as well as further classifying landscape images (Vailaya, Jain & Zhang, 1998). Semantic modelling, as proposed by Vogel & Schiele (2007), classifies local image regions into semantic concept classes, such as water, trees and rocks, to increase results.

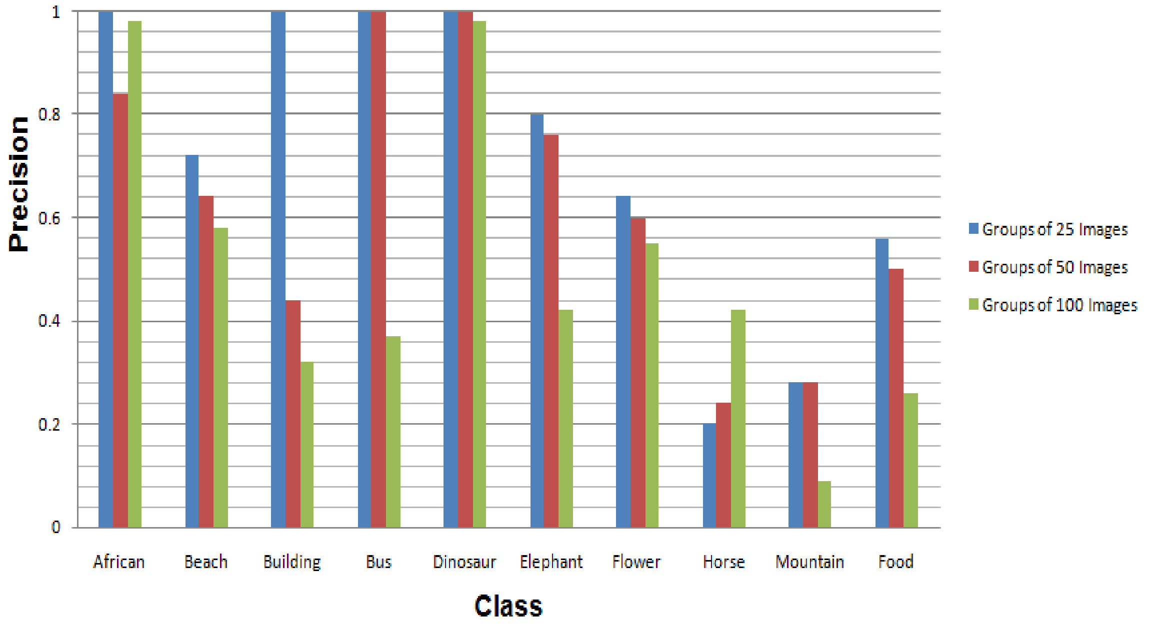


Figure 6.104: Comparison of Precision values

The African and Dinosaur classes produced very high precision within all three sets, while the Buses class performed significantly in the 25 image and 50 image groups but lower in the 100 image group. The overall precision for the 25 image and 50 image groups are 0.72 and 0.63 respectively.

Within the analysis of the results it is vital to include a discussion regarding the error rate of the chosen Corel Wang database. Deselaers (2003) stated that error rate is also an evaluation measure that represents the percentage of images that are incorrectly identified from a class. Therefore, the equation of error rate is defined as follows:

$$ER = 1 - P(1) \quad \text{Equation 6.2}$$

Where

P(1) is the average precision for the first results over the performed queries

Table 6.7 shows the tested error rate on the sets of image groups retrieved, and its computation is based on equation 6.2. As the number of images increases, so does the error rate. Therefore, the error rate is directly proportional to the number of images within a group. This system is reliable for a sizable database on a set of groups at 14 images. As the number of sets of image groups grows, a parallel database will be required to ease the workload of only one database.Therefore, it is advisable to divide sets of groups into smaller tasks then store them in different databases to reduce error rate in this proposed work.

Table 6.7: Error rate of the system

Category of sets	Error rate (ER)
Set of group of 14 images	ER_14 = 1 - 1 = 0 or 0%
Set of group of 25 images	ER_25 = 1 - 0.72 = 0.28 or 28%
Set of group of 50 images	ER_50 = 1 – 0.63 = 0.37 or 37%
Set of group of 100 images	ER_100 = 1 – 0.497 = 0.503 or 50.3%

6.7 Conclusion

In conclusion, an image classification, storage and retrieval system for 3U CubeSat was designed and developed using a generated graphical user interface in Matlab. This interface allowed users to select and view a list of images and search the database, provided that database images features are extracted.

A database was designed to contain all images feature vectors and different tables were created to assist in the storing of the feature vectors. Distance measure formulae were used to compare stored images by retrieving values from the desired tables.

The developed software was subsequently tested, as well as the extracted features and image metadata. The system performance and evaluation was carried out and the proposed algorithms were evaluated. Obtained results were also compared to existing systems and showed increased performance and precision.

Lastly, tests were done to verify the accuracy of sets of groups of 25, 50 and 100 images. Each set contained ten classes. The total number of images in each set equals 250, 500 and 1000 images.

CHAPTER 7

CONCLUSION

7.1 Introduction

The purpose of this project is to design an image classification, storage and retrieval system for a 3U CubeSat. In Chapter Two, various types of feature extraction methods such as colour, shape and texture were studied with the main objective to classify and extract image content. In Chapter Three and Four, the two categories of image classification namely supervised and unsupervised classification were studied in detail. This chapter reflects on the aim and objectives of the project in Section 7.2, the system testing and evaluation in Section 7.3 and provide future work and recommendations in Section 7.4.

7.2 Aims and Objectives

The main focus of this study was to develop and design an image classification, storage and retrieval system for a CubeSat. This objective was achieved by providing an image classification, storage and retrieval system for a CubeSat that is able to classify images according to date and time, colour, shape and texture.

In terms of the stated sub-objectives:

7.2.1 To study and analyze space and Earth images and study methods to identify specific features.

The literature review on how to extract relevant features from an image were studied in Chapter Two. Techniques according to colour feature, shape feature and texture feature were investigated in order to identify features in an image. Identified Colour feature techniques included Colour Histograms, Colour Coherence Vector, Colour Correlograms and Colour Moments, and were studied in detail. Shape feature techniques such as boundary and region based shapes and texture feature techniques for instance Wavelet Transform, Gabor Wavelet, Tamura Feature, GLCM and LBP were studied.

7.2.2 To select and propose appropriate image processing techniques to process and extract features in an image.

From the detailed literature reviews, a suitable image processing technique to process and extract feature in an image were put forward. The Colour Histogram was chosen for colour features as it is a a global statistical feature that symbolizes the distribution of colours in an image. Colour histogram is robust, fast, require low storage and is easy to implement.

For the shape features, the chosen method was Mathematical Morphology, as it is . used to investigate the interaction between an image and a certain chosen structuring element by using the basic operations of erosion and dilation. This basic operation assists to identify objects which can be contained in an image. Shape features are vital in content based image retrieval, as they satisfy the requirement that two similar shapes in different images can be compared as they tend to be close or identical to one another's values.

Within the texture feature the chosen method is GLCM which computes features such as Contrast, Energy, Correlation, and Homogeneity of gray level integer values. The contrast computes a total of local variations present in an image while energy is the sum of squared elements in GLCM. The homogeneity are diagonal distribution of elements in GLCM. Lastly, correlation illustrates how associated a pixel is to its neighbour in the image. Using this features an image can be compared and given a sample image the system will be able to identify what group/classes it might belong to. As GLCM rely on distance and direction of pixels in an image and four directions were used.

7.2.3 To design and propose an image classification scheme suited for CubeSat images.

A design and appropriate image classification techniques suitable for satellite images was designed in Chapter Five. The proposed design has broadly five stages namely input image stage, feature extraction stage, compare, rank and display stage and database storage stage. In the input stage an appropriate image is chosen from a list of images. Image noise is then removed using median filter method and image quality is improved. Feature extraction stage is whereby all suitable features are extracted from the image using Colour, Shape, Texture, Date and Time extraction methods. In the comparison stage, rank and display stage comparison is made using the distance measure of a query and stored images within the database. When values from the feature extraction method are used and are close to one another, images can be retrieved, ranked and displayed. The database storage stage: this is where all features extracted from an image are kept.

7.2.4 To classify images using image classification techniques (i.e., texture classification), once received from 3U CubeSat to ground station.

Upon completion of 3U CubeSat to be launched into the desired orbit, it is satellite images will be downlinked and received on the built bellville ground station, where images will be store in the database. However, before images can be stored, appropriate feature extraction techniques will be used to compute, classify and store features in the database. Image features will be stored according to the date and time it were taken. The colour histogram and colour occurrence frequency pixels value will be computed and stored in the table called color-feature-table. Shape mathematical morphology parameters such as area, perimeter and metric will be computed so that an object in an image can be compared. Texture techniques parameters for instance as energy, correlation, homogeneity and contrast values will be computed and used to classify similar images which may belong in the same group.

7.2.5 To design a storage and retrieval system

The objective of designing a storage and retrieval system was achieved in Chapter Six. In Section 5.4.7 and Section 6.4 a database was designed and different tables were created to store various computed features being extracted from the images.

7.2.6 To study how images can be ranked and retrieved from a large database

In the proposed design discussed in Chapter Five, distance measure techniques were studied in Chapter Two (Section 2.7) and used to compare two images. The query image are compared against images in the database, and matches was obtained. To compute the ranking of the retrieved images, the distance measure values are used to find maximum or minimum values relating to each image. Sorting algorithms are then used to swap the highest value to lowest value.

7.2.7 To study techniques and evaluate the performance of an image retrieval system

In image retrieval systems, the performance and evaluation are vital for checking efficiency and effectiveness of the system. In this project, the precision and recall techniques were studied and used to evaluate the proposed system as discussed in Chapter Six (Section 6.6). Each proposed algorithm namely colour, shape, texture and date classifier were tested, evaluated and compared to other systems algorithms. The integrated proposed system when compared to other systems outperforms with an average precision of 100%. The proposed system is based on classifying, storing and retrieving of satellite images and can be used in high resolution applications.

7.2.8 To test, implement and integrate the proposed system for the CubeSat environment

Through experiments carried out in Chapter Six, the proposed design (Section 5.4) was implemented and tested (see Section 6.1 to Section 6.6). Each unit component of the proposed design was build and and different software tools were constructed in order to test functionality of the feature extraction, storing and retrieval of the system.

The integrated proposed system was implemented, tested and evaluated in Section 6.6.1(d).

7.3 System Testing and Evaluation

System testing was carried out by interfacing all unit components such as colour feature extractor tool, shape feature extractor tool, texture feature extractor tool and image metadata tool. Each component was tested as were discussed in section 6.5.1. The system's performance was evaluated using test set of Wang database, Ikonos, Quick Bird and Masat-1 high resolution images and various cloud images found in Google (see Appendix B for some of the used images). To evaluate the proposed system with other systems, the Wang database was used as seen in Section 6.6. Each group was reduced from one hundred image to thirty images from which feature vectors were extracted and stored. The accuracy of retrieval depends on selecting one query sample from the 10 groups.

7.4 Future Recommendation

Future research include the following:

- Graphical user interface to include other features such as colour moment, colour correlograms, local binary pattern and colour coherence vectors. Texture features such as wavelet transforms and Gabor filtering to be implemented
- Image classification should consider an image of geolocation or geotagged using altitude and longitude of the place where the image was taken.
- Design and implement standalone image classification system using other high level languages such as Python, C++ and Java.
- Extend the designed system to support different database systems.

References

AAU CubeSat. 2012. OBC Computer. http://www.space.aau.dk/cubesat [20 August 2012].

Abdalla, O. A., Zakaria, M. N., Sulaiman, S. & Ahmad, W. F. 2010. A comparison of feedforward back propagation and radial basis artificial neural networks: A Monte Carlo study. *Proceedings of the International Symposium on Information Technology (ITSim)*, Kuala Lumpur Convention Centre, Kuala Lumpur, Malaysia, 15-17 June 2010.

Abdelrahaman, A. A. & Abdallah, M. E. 2013. K-nearest neighbor classifier for signature verification system. *Proceedings of the International Conference on Computing, Electrical and Electronics Engineering (ICCEEE),* Khartoum, Sudan, 26-28 August 2013.

Achary, T. & Tsai, P. S. 2005. *JPEG 2000 Standard for image compression: concepts,algorithms and VLSI architectures.* New Jersey: Wiley.

Afifi, A. J. 2011. Image retrieval based on content using color feature. M.Tech Thesis, Islamic University of Gaza, Palestine.

Agarwal, B. B. & Tayal, S. P. 2007. *Software Engineering.* New Delhi: Laxmi.

Agarwal, B. B., Tayal, S. P. & Gupta, M. 2010. *Software Engineering and testing.* London: Jones & Bartlett.

Aggarwal, C. C. & Reddy, C. K. 2014. *Data clustering: algorithms and applications.* Boca Raton: CRC Press.

Ahmad, A. & Sufahari, S. F. 2012. Analysis of landsat 5 TM data of Malaysian land covers using isodata clustering technique. *Proceedings of the IEEE Asia-Pacific Conference on Applied Electromagnetics (APACE), Malaka, Malaysia,11-13 December 2012.*

Ahmadian, A. & Mostafa, A. 2003. An efficient texture classification algorithm using Gabor wavelet. *Proceedings of the 25th Annual International IEEE Conference in Medicine and Biology Society (EMBS), Cancun, Mexico, 17-21 September 2003.*

Ahmed, H. Z. 2011. Design and implementation of car plate recognition system for Ethiopian car plates. MSc. Thesis, Addis Ababa University, Ethiopia.

Aldayel, M. S. 2013. K-nearest neighbor classification for glass identification problem. *Proceedings of the International Conference on Computer Systems and Industrial Informatics (ICCSII),* American University of Sharjan, Sharjah, United Arab Emirates, 18-20 December 2012.

Al-Fayadh, A. H., Mohammed, H. R. & Al-shimsah, R. S. 2012. Gabor wavelet transform in image compression. *Journal of Kufa for Mathemaics and Computer, 1*(6):107-113, December.

Amlani, R. D. 2013. Comparison of different SDLC models. *International Journal of Computer Application & Information Technology, 2*(1): 4-8, January.

Amuthajanaki, B. & Jayalakshmi, K. 2013. A hierarchical divisive clutering based multi-view point similarity measure for document clustering. *International Journal of Advances in Computer Science and Technology, 2* (8):155-159.

Anthony, G., Gregg, H. & Tshilidzi, M. 2007. *Image classification using SVMs: One-Against-One Vs One-Against-All*. *Proceedings of the 28th Asian Conference on remote Sensing (ACRS)*, Kuala Lumpur, Malaysia, 12-16 November 2007.

Aslan, A. R., Sofyali, A., Umit,E., Tola, C.,Oz, I. & Gulgonul, S. 2011. TURKSAT-3USAT: A 3U communication CubeSat with passive magnetic stabilization. *Proceedings of fifth international Conference on Recent Advances in Space Technology (RSAT), Istanbul, 9-11 June 2011.*

Asubam, W. B. 2011. Improve median filtering algorithm for the reduction of impulse noise in corrupted 2D greyscale images. MSc Thesis, Kwame Nkrumah University of Science and Technology, Ghana.

Atwan, A. 2012. 3D Isodata and region growing technique for segementation a true color visible human dataset. *Proceedings of the 8th International Conference on Informatics and Systems and Computational Intelligence and Multimedia Computing Track*, Cairo University, Giza, Egypt, 14 -16 May 2012.

Babuscia, A., Corbin, B. & Clem, R. J. 2013. CommCube 1 and 2: A CubeSat series of missions to enhance communication capabilities for CubeSat. *Proceedings of the 2013 IEEE Aerospace Conference, Big Sky, Montana, 2-9 March 2013.*

Banfi, F. 2000. Content based mage retrieval using hand drawn sketches and local features: A study on visual dissimilarity. PhD Thesis, University of Fribourg, Switzerland.

Baoli, L., Shiwen, Y. & Qin, L. 2003. An improved k-nearest categorization. *Proceedings of the 20th International Conference on Computer Processing of Oriental Languages.* Shenyang,China.

Bataineh, M. H. 2012. Artificial neural network for studying human performance. M.Tech. Thesis, University of Iowa, United States.

Benoso, B.L. & Nazuno, J.F. 2008. Cellular Mathematical Morphology. *Proceedings of the 6th Mexican International on artificial intelligence*, Aguascalientes, Mexico, 4-10 November 2004.

Bezold, M. 2013. An attitude determination system with mems gryroscope drift compensation for small satellites. MSc. Thesis, University of Kentucky, Kentucky.

Bhardwaj, R. & Vatta, S. 2013. Implementation of ID3 algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(6): 845-851, June.

Bhatia, N. & Vandana. 2010. Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security, 8* (2): 302-305.

Bidgoll, H. 2004. *The Internet Encyclopedia.* New Jersey: Wiley.

Bovik, A. 2000. *Handbook of image and video processing.* San Diego: Academic Press.

Bovik, A. 2005. *Handbook of Image and Video Processing.Second Edition.* Amsterdam: Elsevier Academic Press.

Bruha, I. 2000. From machine learning to knowledge discovery: Survey of pre-processing and post processing. *Intelligent data analysis, 4* (3-4): 363-374.

Budd. 2007. Content based image retrieval using sketched input. BSc. Thesis, University of Bath, United Kingdom.

Burt, R. 2011. Distributed electrical power system in CubeSat applications. MSc. Thesis, Utaha State University, Logan.

Cape Peninsula University of Technology (CPUT). 2013. Nanosatellite Technology at CPUT. http://active.cput.ac.za/ [10 November 2013].

Carreiras, J.M.B., Pereira, J.M.C., Campagnolo, M.L. & Shimabukuro, Y.E. 2006. Assessing the extent of agriculture/pasture and secondary succession forest in the Brazilian legal Amazon using spot vegetation data. *Remote sensing of Environment,* 101:283-298, December.

Castelli, V. 2001. Multidimensional indexing structures for content based retrieval. *Computer Science* RC 22208(98723):1-65, February.

Chan, C. H. 2008. Muli-scale local binary pattern histogram for face recognition. PhD Thesis, University of Surrey, England.

Chiu, C.Y., Lin, H.C. & Yang, S.N. 2001. Texture retrieval with linguistic descriptions. *Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, Beijing, China, 24-26 October 2001.

Chocano, E. M. 2004. *A comparative study of iterative prototyping vs waterfall process applied to small and medium sized software project.* Master's Thesis, Massachusetts Institute of Technology, Massachusetts.

Christiyana, C. C. & Rajamani, V. 2012. Comparison of local binary pattern variants for ultrasound kidney image retrieval. *International Journal of Advanced Research in Computer Science and Software Engineering, 2*(10):224-228.

Chun-Yu, N., Shu-Fen, L. & Ming, Q. 2009. Research on removing noise in medical images based on median filter method*. Proceedings of the IEEE symposium on Information Technology(IT) in Medicine and Education, Ji'nan, China, 14-16 August 2009.*

Clarke, B., Fokoue, E. & Zhang, H. H. 2009. *Principle and theory for data mining and machine learning.* London: Springer.

Clyde Space. 2012. CubeSat electrical power subsystems. http://www.clyde-space.com/ [11 November 2012].

Cuadra, C.A. 2006. Annual review of information science and technology. Medford, NJ: Information Today.

Czibula, I. G. & Czibula, G. 2008. Clustering based automatic refactorings identification. *Proceedings of the 10th International Symposium on Symbolic and numeric algorithms for scientific computing, Timisoara, Romania, 26-29 September 2008.*

Dabas, P. & Chaudhary, R. 2013. Survey of network intrusion detection using k-Means algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(3): 507- 511, March.

Dellaert, F. 2002. The expectation maximization algorithm. *Georgia Institute of Technology*, 1-7.

Deotale, N., Vaikole, S. L. & Sawarkar, S. D. 2010. Face recognition using Artificial neural neworks. *Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, 26-28 February 2010.*

Deselaers, T. 2003. Features of Image Retrieval. Diplomarbeit im Fach Informatik, Rheinisch-Westfälische Technische Hochschule Aachen, Germany.

Devi, V. S. & Murty, M. N. 2002. An incremental prototype set building technique. *Pattern Recognition, 35* (2): 505 - 513.

Dhasal, P., Shrivastava, S. S., Gupta, H. & Kumar, P. 2012. An optimized feature selection for image classification based on SVM-ACO. *International Journal of Advanced Computer Research , 2* (3):123-128, September.

Dogdas, T. & Akyokus, S. 2013. Documents clustering using GIS visualizing and EM clustering method. *Procedings of the IEEE International Symposium on Innovations in Intelligent Systems and Applications, Albena, Bulgaria, 19 -21 June 2013.*

Dong-Cheng, S., Lan, X. & Ling-Yan, H. 2007. Image retrieval using color and texture features. *The Journal of China Universities of Posts and Telecommunications, 14:*1-6, October.

Edvardsen, S. 2006. Classification of images using CBIR distance measures and genetic programming: An evolutionary experiment. MSc. Thesis, Norwegian University of Science and Technology, Norway.

ElAlami, M. E. 2014. A new matching strategy for content based image retrieval system. *Applied Soft Computing, 14*:407-418.

Elomaa, T. & Rausu, J. 1999. General and efficient multisplitting of numerical attributes. *Machine Learning, 36:* 201-244, May.

Esposito, F., Malerba, D. & Semeraro, G. 1997. A comparative analysis of methods for pruning Decision Tree. *IEEE Transactions of Pattern analysis and Machine Intelligence, 19* (5): 476- 491.

Eva, H., Carboni, S., Achard, F., Stach, N., Durieux, L. & Faure, J.F. 2010. Monitoring forest areas from continental to territorial levels using a sample of medium spatial resolution satellite imagery. *ISPRS Journal of photogrammetry and remote sensing*, 65(2):191-197.

Fang, F.M. 1997. An analytical study on image databases. MSc. Thesis, Massachusetts Institute of Technology, United States.

Fang, Z. & Yulei, M. 2012. Medical image processing based on mathematical morphology. *Proceedings of the 2nd International Conference on computer applications and system modelling,* Shanxi, Taiyuan, 22-24 October 2010.

Feng, D. D. 2008. *Biomedical information technology*. London: Academic Press.

Feng, D., Siu, W.C. & Zhang, H.J. 2003. (Eds). *Multimedia information retrieval and management*: Technological fundamentals and applications. Berlin: Springer.

Figueiredo, M., Botelho, S., Drews, P. & Haffele, C. 2012. Self organizing mapping of robotic environments based on neural networks. *Proceedings of the Brazilian Symposimum on neural networks,* Curitiba, Parana, Brazil, 20- 25 October 2012.

Flusser, J. 2005. Moment Invariants in image analysis. *World Academy, Engineering and Technology, 11*:376-381.

Frank, E. 2000. Pruning Decision Trees and lists. PhD Thesis, University of Waikato, New Zealand.

Freitas, A. A. 2002. *Data mining and knowledge discovery with evolutionary algorithms.* Berlin: Springer.

French South Africa Institute of Technology (FSATI). 2012. CubeSat projects at CPUT .http://active.cput.ac.za/fsati [11 July2012].

Fuentealba, C. & Charpentier, P. 2004. The K-nearest neighbor method for automatic identification of wood products. *Proceedings of the 14th International Conference on Electronics, Communications and Computer (CONIELECOMP'04), Veracruz, Mexico, 16-18 February 2004..*

Fuertes, J., Lucena, M., Perez, N. & Martinez, J. 2001. A scheme of color image retrieval from databases. *Pattern recognition letters, 22*:323-337.

Futrell, R. T., Shafer, D. F. & Shafer, L. I. 2002. *Quality software project management.* New Jersey: Prentice Hall PTR.

Gejun, Z., Changsheng, M. & Feng, X. 2010. The K-nearest neighbor fast searching algorithm of scattered data. *Proceedings of the International Conference on Future Information Technology and Management Engineering(FITME),* Changzhou, China, 9-10 October 2010.

Geng, X., Liu, T. Y. & Qin, T. 2008. Query dependent ranking using K-nearest neighbor. *Proceedings of the 31st Annual Internationala ACM SIGIR Conference on Research and development in Informational retrieval,* Singapore, 20-24 July 2008.

Ghorpade, J. & Agarwal, S. 2011. SOM and PCA approach for face recognition: A survey. *International Journal of Computer Trends and Technology (IJCT)*: 1-6,March-April.

Gloe, T. 2012. Forensic analysis of ordered data structures on the example of JPEG files. *IEEE International Workshop on Information Forensics and Security (WIFS)*, Costa Adeje, Tenerife, Spain, 2-5 December 2002.

Gonzalez, R. C. & Woods, R. E. 2002. *Digital image processing.* New Jersey: Prentice Hall.

Gopi, E. S. 2007. *Algorithms collections for digital signal processing applications using Matlab.* Dordrecht: Springer.

Gotlieb, C. C. & Kreyszig, H. E. 1990. Texture descriptors based on co-occurrence matrix. *Computer vision, graphics and image processing, 51*(1): 70-86.

Gu, H., Su, G.D. & DU, S. 2004. Fuzzy and isodata classification of face contours. *Proceedings of the third International Conference on Machine Learning and Cybernetics*, Shanghai, China, 26-29 August.

Gualtieri, J. A. & Cromp, R. F. 1998. Support Vector Machine for hyperspectral remote sensing classification. *Proceedings of the 27th Advances in Computer Assisted Recognition (AIPR) Workshop,* Washington, DC, 14 -16 October 1998.

Guan, H., Antani, S., Long, L. R. & Thoma, G. R. 2009. Bridging the semantic gap using ranking svm for image retrieval. *Proceedings of the 6th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, Boston, Massachusetts, 28 June – 1 July 2009.

Gupta, G.K. 2011. *Introduction to data mining with case studies.* Second edition. New Delhi: PHI Learning Private Limited.

Hajek, M., Dezortova, M., Materka, A.. & Lerski, R. 2006. *Texture analysis for magnetic resonance imaging.* Prague: Med4publishing.

Han, J. & Kamber, M. 2001. *Data mining: concepts and techniques.* London: Academic Press & Morgan Kaufman.

Han, J., Kamber, M. & Pei, J. 2012. *Data mining: concepts and techniques.* Waltham: Elsevier.

Hang, C. C. 2004. *Using biased support vector machine in image retrieval with self organizing map.* MSc. Thesis, Chinese University of Hong Kong, China.

Haralick, R. M., Shanmugam, K., & Dinstein, I. 1973. Textural Features for image classification. *IEEE Transactions on systems, man and cybernetics,* SMC-3(6): 610-621, November.

Hardikar, S., Shrivastava, A., & Choudhary, V. (2012). Comparison between ID3 and C 4.5 in contrast to IDs. *VSRD International Journal of Computer Science and Information Technology , 2* (7): 659-667.

He, D. 2010. Three new methods for color and textures based image matching in content based image retrieval. PhD Thesis, Dalhousie University, Canada.

He, G. & Zhu, Y. 2011. A self-adaptive EM-HMRF image fusion algorithm based on non-homogeneous class and direction. *Proceedings of the APSIPA Annual Summit and Conference,* Xi'an, China, 19-21 October 2011.

He, Z. & Zhan, Z. 2011. Digital halftoning based on k-Means clustering. *Proceedings of the 7th IEEE Conference on Industrial Electronics and Applications (ICIEA),* Singapore, 18-20 July 2012.

Hong, P., Tian, Q. & Huang, T. S. 2000. Incorporate support vector machines to content based image retrieval with relevant feedback. *Proceedings of the IEEE International Conference on Image processing(ICIP).* Vancouver,Canada,10-13 September 2000.

Hongli, X., De, X. & Yong, G. 2004. Region based image retrieval using color coherence region vectors. *Proceedings of the 7th International Conference on Signal Processing (ICSP'04),* Beijing, China, 31 August- 4 September 2004.

Hongping, H. & Yanping, B. 2011. A kind of license plate location based on mathematical morphology and edge detection. *Proceedings of the International conference on Electronic and Mechanical Engineering and Information Technology, Harbin, China, 14 August 2011.*

Horais, B. & Twiggs, R. 2001. Re-injecting innovation into the space test process. *Proceedings of the 15th Conference on small satellites, Logan, Utah, 13-16 August 2001.*

Hu, M. K. 1962. Visual pattern recognition by moment invariants. *IRE Transaction information theory, 8 :*179-187.

Huang, J., Kumar, R. S., Mitra, M. & Zhu, W. I. 2002. *Image sub region query using color correlograms.* Cornell Research Foundation. http://www.google.com/patents/US6430312.

Huang, Z., & Leng, J. 2010. Analysis of Hu's moment Invariants on Image scaling and rotation. *Proceedings of the 2nd International Conference om Computer Engineering and Technology (ICCET), Chengdu, China, 16-18 April 2010.*

Hui, D. 2009. An improved method of CBR retrieval based on self organizing map. *Proceedings of the IEEE International Conference on Inelligent Computing and Intelligent Systems,* Shanghai, China, 20-22 November 2009.

Hussain, C.A., Rao, D.V. & Praveen, T. 2013. Color histogram based retrieval. *International Journal of Advanced Engineering Technology, 4(3):63-66,September.*

Imandoust, S. B. & Bolandraftar, M. 2013. Application of K-nearest (KNN) approach for predicting economic events: Theoretical Background. *Journal of Engineering Research and Application, 3*(5): 605-610.

Indian Institute of Technology Kanpur (IITK). 2013. Jugnu Subystems. http://iitk.ac.in/me/jugnu/subsystem.html [13 November 2013].

Ismailoglu, N., Benderli, O., Korkmaz, I., Kolcak, T. & Tekmen, Y. C. 2002. A real time image processing subsystem: Gezgin. *Proceedings of the 16th Annual AIAA/USU Conference on small satellites, Logan Utah, 12-15 August 2002.*

Ismailoglu, N., Benderli, O., Yesil, S., Sever, R., Okcan, R. & Oktem, R. 2005. Gezgin 2: An advanced image processing subsystem for earth observing small satellites. *Proceedings of the 2nd International Conference on Recent Advances in Space Technologies,* Instanbul, 9-11 June 2005.

Jahangeer, N.B. & Baichoo, S. 2013. Face recognition using chain codes. *Journal of Signal and Information Processing (JSFP)*, 4:154-157.

Jain, S. 2013. Brain cancer classification using glcm based feature extraction in artificial neural network. *International Journal of Computer Science & Engineering Technology(IJCSET), 4*(7):966-970, July.

Jalote, P. 2008. *A concise introduction to software engineering.* London: Springer.

Jamil, N., Iqbal, S. & Iqbal, N. 2001. Face recognition using neural networks. *Proceedings of the IEEE International Multi Topic Conference (IEEEINMIC), Lahore University of Management Sciences, Pakistan, 28 – 30 December 2001.*

Jamwal, D. 2010. Analysis of software development models. *International Journal of Computer Science and Technology* (IJCST)*, 1*(2):61-64, December.

Jiang, C., Zhao, W., Xu, W. & Meng, X. 2009. Research of Fingerprint recognition. *Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC'09), Changdu, China, 14 December.*

Jin, A. H., Chekima, A., Dargham, J. A. & Fan, L. C. 2002. Fingerprint identification and recognition using backpropagation neural network. *Proceedings of Student Conference on Research and Development, Sah Alam, Malaysia,17- 17 July 2002.*

Jin, Y.A. & Xiao, F. 2013. Eliminating error accumulation in hierarchical clustering algorithms. *Proceedings of the 4th International Conference on Emerging Intelligent Data and web Technologies*, Xi'an, China, 9-11 September 2013.

Jivani, A. G. 2013. The novel k-nearest neighbor algorithm. *Proceedings of the International Conference on Computer Communication (ICCCI-2013),* Coimbatore, India, 4-6 January 2013.

Junfeng, Z. & Leping, X. 2010. The fault diagnosis of marine engine cooling system based on artiificial neural network. *Proceedings of the 2nd International Conference on Computer and Automation Engineering, Singapore, 26- 28 February 2010.*

Kadir, A., Nugroho, L. E., Susanto, A. & Santosa, P. I. 2011. Neural network application on foliage plant identification. *International Journal of Computer Applications, 29* (9):15-22.

Kaghyan, S. & Sarukhanyan, H. 2012. Activity recogntion using k-nearest neighbor algorithm on smartphone with tri-axial accelerometer. *International Journal Information models and analyses , 1*: 146-156.

Kakar, P., & Sudha, N. 2012. Verifying temporal data in geotagged images via sun azimuth estimation. *IEEE Transaction on Information Forensics and Security*, 7(3):1029-1039.

Kale, K. V., Mehrotra, S. C. & Manza, R. R. 2007. *Advances in computer vision and information technology.* New Delhi: I.K International Publishing House.

Kalman, A. & Reif, A. 2008. MISC™: A novel approach to low cost imaging satellites. Proceedings of 22 second annual AIAA/USU Conference on small satellite, Logan, Utah, USA, August 10.

Kan, S. H. 2003. *Metrics and models in software quality engineering.* Boston, MA: Pearson Education.

Kan, S. K. 2008. *Metrics and models in software quality engineering.* Second Edition, Boston: Pearson Education.

Kang, K. H., Yoon, Y. I., Choi, J. S., Koo, H. & Choi, J. H. 2007. Additive texture information extraction using color coherence vector. *Proceedings of the 7th World Scientific Engineering Academy and Society (WSEAS) International Conference on Multimedia systems and signal processing*. HangZhau, China, 17 April 2007.

Kantardzic, M. 2011. *Data Mining: Concepts, models, methods and algorithms.* New Jersey: Wiley and IEEE Press.

Kaplan, C. 2006. *Leo Satellites: attitude determination and control components: some linear attitude control techniques.* MSc.Thesis, Middle East Technical University, Turkey.

Karthikeyan, T., Manikandaprabhu, P. & Nithya, S. 2014. A survey on text and content based image retrieval system for image mining. *International Journal of Engineering Research & Technology (IJERT), 3*(3):509-512, March.

Karvonen, M. 2008. Using mathematical morphology for geometric music retrieval. MSc. Thesis, University of Helsinki, Finland.

Kascheck, R., Kop, C., Steinberger, C. & Fliedl, G. 2008. *Information systems and e-business technologies.* Berlin: Springer.

Kashef, R. & Kamel, M. S. 2007. Cooperative partitional-divisive clustering and its application in Gene expression analysis. *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengeering*, Boston, Massachusetts, USA, 14-17 October 2007.

Kee, E., Johnson, M. K. & Farid, H. 2011. Digital image authentication from JPEG headers. *IEEE Transactions on Information Forensics and Security, 6*(3):1066-1075, September.

Kehtarnavaz, N. & Gamadia, M. 2005. *Real time image and video processing: From research to reality.* Dallas: Morgan & Claypool.

Khursid, K., Mahmood, R. & Islam, Q. U. 2013. A survey of camera modules for CubeSats: Design of imaging payload of ICUBE-1. *Proceedings of the 6th International Conference on Recent Advances in Space Technologies (RAST),* Istanbul, Turkey, 12-13 June 2013.

Kim, W., Lee, H. K. & Yoon, K. 2008. Hierarchial adult image rating system. *LNCIS, 345*: 894-899.

Klofas, B. & Leveque, K. 2013. A survey of CubeSat communication systems: 2009-2012. *Proceedings of the 10th Annual CubeSat Developers Workshop,* CalPoly, San Luis Obispo, 24-26 April 2013.

Kotsiantis, S. B. 2007. Supervised Learning: A review of classification techniques. *Informatica* 31: 249 - 268, July.

Krishnamachari, S. & Abdel-Mottaleb, M. 1998. A scalable algorithm for image retrieval by color. *Proceedings of International Conference on image processing (ICIP),* Chicago, Illinois, USA, 4-7 October 1998.

Krishnamurty, N. 2008. *Dynamic modelling of CubeSat project move.* M.Tech. Thesis, Lulea University of Technology, Sweden.

Kumar, D. A., & Esther, J. 2011. Comparative study on CBIR based by color histogram, Gabor and wavelet Transform. *International Journal of Computer Appplications, 17*(3): 37-44, March.

Kurhe, A. B., Satonka, S. S. & Khanale, P. B. 2011. Color matching of images by using Minkowski form distance. *Global Journal of Computer Science and Technology, 11*(5): 86-89, April.

Lalhmingliana, Bhattacharyya, D. K. & Sing, K. R. 2013. Fingerprint indexing and verification. *International Journal of Computer Science Engineering and Information Technology, 3*(2): 167-176, June.

Laplante, P. & Neill, C. 2003. A class of kalman filters for real time image processing. *The International Society for Optics Engineering (SPIE), Realtime imaging VII (5012), Santa Clara, CA, April 2003.*

Larranaga, P. & Lozano, J. A. 2011. *Estimation of distribution algorithms: a new tool for evolutionary computation.* Norwell, Massachusetts : Kluwer Academic Publishers.

Larson, J. W. & Wertz, J. R. 1999. *Space mission analysis and design.* London: Kluwer Academic Publishers.

Latha, Y. M., Jinaga, B. C. & Reddy, V. K. 2007. Content based color image retrieval via wavelet transforms. *International Journal of Computer Science and Network Security, 7* (12):38-45, December.

Lavoie, T. & Merlo, E. 2012. An accurate estimation of the levenshtein distance using metric trees and manhattan distance. *The 6th International workshop on Software Clones (IWSC), Zurich, Switzerland, 4 June 2012.*

Lee, C. H. & Fu, Y. H. 2008. Web usage mining based on clustering of browsing features. *Proceedings of 8th International Conference on Intelligent Systems Design and Applications (ISDA), Kaohsiung, Taiwan, 26-28 November 2008.*

Lehman, T. J. & Sharma, A. 2011. Software development as a service: agile experiences. *Proceedings of the Annual SRII Conference, San Jose, California, USA, 29 March - 2 April 2011.*

Li, C., Li, J., Fu, B. & Yang, X. 2012a. Fingerprint verification based on DFB and Hu Invariant moments. *Journal of computational information systems, 8*(4):1407-1414.

Li, J., Shao, B., Li, T. & Ogihara, M. 2012b. Hierarchical co-clustering: A new way to organize the music data. *IEEE Transactions on Multimedia*, 14(2): 471 - 481, April.

Liang, J. Z. 2003. Chinese web page classification based on self organizing mapping neural networks. *Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'03).*

Lin, C. H., Chen, R. T., & Chan, Y. K. 2009. A smart content based image retrieval sysem based on color and texture feature. *Image and vision computing, 27*: 658-665.

Lin, H. C., Chiu, C. Y. & Yang, S. N. 2002. Texture analysis and description in linguistic terms. *Proceedings of the 5th Asian Conference on Computer Vision (ACCV2002).* Melbourne, Australia, 25 January 2002.

Lin, H. C., Chiu, C. Y., & Yang, S. N. 2003. Finding textures by textual descriptions, visual examples and relevance feedbacks. *Pattern recognition letters 24* :2255 - 2267.

Liping, Z., & Wensheng, G. 2009. Self organizing maps neural networks on parallel cluster. *Proceedings of the 1st International Conference on Information Science and Engineering (ICISE), Nanjing, China, 26 - 28 December 2009.*

Liu, B. & Liew, S. 2007. Texture retrieval using grey level co-occurrence matrix for Ikonos panchromatic images of earthquake in Java 2006. *Proceedings of the IEEE International Geoscience and remote sensing symposium (IGARSS), Barcelona, 23-28 July 2007.*

Liu, F., Lin, C.X., Cui, P.Y. & Dong, T. 2007. Palmprint recognition based isodata clustering algorithm. *Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition, Beijing, China, 2-4 November 2007.*

Long, F., Zhang, H. & Feng, D. 2002. Fundamentals of content based image retrieval. In Feng, D., Siu, W.C. & Zhang, H. (Eds). *Multimedia Information Retrieval and Management Technology Fundamentals and Applications*, Berlin: Springer.

Luckey, T. & Phillips, J. 2006. *Software project management for dummies.* Hoboken: Wiley.

Madsker, L. R. & Jain, L. C. 2000. *Recurrent neural networks: Design and applications.* New York: CRC Press.

Mahanti, R., Neogi, M. S. & Bhattacherjee, V. 2012. Factors affecting the choice of software life cycle models in the software industry: An empirical study. *Journal of Computer Science , 8* (8): 1253-1262.

Maheswary, P. & Srivastava, N. 2009. Retrieval of remote sensing images using colour & texture attribute. *International journal of Computer Science and Infomation Security* (IJCSIS), 4(1-2):1-5.

Maheshwari, S. & Jain, D. C. 2012. A comparative analysis of different types of models in software development lifecycle. *International Journal of Advanced Research in Computer Science and Software Engineering, 2*(5): 285-289, May.

Mahmood, A. M.& Kuppa, M. R. 2012. A novel pruning approach using expert knowledge for data-specific pruning. *Engineering with computers, 28:*21-30.

Makarov, A. 2012. Fingerprint quality estimation using self organizing maps. MSc.Thesis, Technical University of Denmark, Denmark.

Marciniak, J. J. 2001. *Encyclopedia of software engineering*. Second Edition. New York: John Wiley & Sons.

Marques, O. 2011. *Practical image and video processing using Matlab.* New Jersey: Wiley.

Masat-1. 2012. Masat-1 first photographs taken during filght. http://cubesat.bme.hu/data/sajtokozlemeny/2012_03_14_Masat-1urfelvetelek.zip[14 November 2013].

Mathur, S. 2012. Artificial intelligence based feature extraction in content based image retrieval. M.Tech. Thesis, Jagan Nath University, Bangladesh.

M-Cubed. 2013. Michigan multipurpose minisat(M3).http://umcubed.org/ [13 November 2013].

Melgani, F. & Bruzzone, L. 2004. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing,* 42(8): 1778 - 1790, August.

Mikhraq, A. K. 2013. Content based image retrieval(CBIR) system based on the clustering and genetic algorithm. MSc. Thesis, Islamic University of Gaza, Palestine.

Ming, H., Wenying, N. & Xu, L. 2009. An improved Decision Tree classification algorithm based on ID3 and the application in score analysis. *Proceedings of the Control and Decision Conference (CCDC),Guilin, 17 -19 June 2009.*

Mishra, A., & Dubey, D. 2013. A comparative study of different software development life cycle models in different scenerios. *International Journal of Advance Research in Computer Science and Management Studies, 1* (5): 64-69,October.

Mitsa, T. 2010. Temporal data mining. Boca Raton: Chapman & Hall/CRC Press.

Mladenovic, V., Lutovac, M. & Lutovac, M. 2012. Automated proving properties of expectation maximization algorithm using symbolic tools. *Telfor Journal , 4* (1). 54-59.

Mohamed, W. N., Salleh, M. M. & Omar, A. H. 2012. A comparative study using expert knowledge for data specific pruning. *Proceedings of the IEEE International Conference on Control system, Computing and Engineering,* Penang, Malaysia, 23-25 November.

Mohammed, A. S., Benyettau, M., Sweeting, M. N., & Cooksley, J. R. 2005. Initial attitude acquisation result of the ALSAT-1 first algerian microsaellite in orbit. *Proceedings of the IEEE Conference on Network, Sensing and Control, 19-22 March 2005.*

Mohapatra, P. K. 2010. *Software Engineering: a life cycle approach.* New Delhi: New Age International.

Mohmood, A. M., & Kuppa, M. R. 2011. A novel pruning approach using expert knowledge for Intelligent in-exact classification. *Proceedings of the 2nd International Conference on Emerging Applications of Information Technology*, Los Alamitos, California, 19-20 February 2011.

Monteiro, I. Q., Queiroz, S. D., Carneiro, A. T., Souza, L. G. & Barreto, G. A. 2006. Face recognition independent of facial expression through som based classifiers. *Proceedings of the 7th International Telecommunications Symposium (ITS2006),* Fortaleza-CE, Brazil, 3-6 September 2006.

Moral, H. 2004. Modeling of activated sudge process by using artificial neural networks. MSc. Thesis, Middle East Technical University.

Mujumdar, A., Masawal, G. & Chawan, P. M. 2012. Analysis of various software process models. *International Journal of Engineering Research and Applications (IJERA), 2*(3): 2015-2021.

Munassar, N. M.A. & Govardhan, A. 2010. A comparison between five models of software engineering. *International Journal of Computer Science, 7*(5): 94-101, September.

Munshi, A. 2010. Skin texture analysis using neural networks for medical diagnosis. M.Tech Thesis, Jadavpur University, Kolkata, India.

Murali , N.V., Raja, K. & Bhanu, K.S. 2013. Content based image search and retrieval using indexing by k-means clustering technique. *International Journal of advanced research in computer and communication engineering,* 2(5):2181-2189, May.

Narayana, M. & Kulkarni, S. 2012. Comparison between euclidean distance metroic and SVM for cbir using level set features. *International Journal of Engineering Science and Technology, 4 (1): 56-66, January.*

Nayak, D. R., Mahapatra, A. & Mishra, P. 2013. A survey on rainfall prediction using artificial neural network. *International Journal of Computer Applications, 72(16):*32-40, June.

Ng, H. P., Ong, S. H., Foong, K. W., Goh, P. S. & Nowinski, W. L. 2006. Medical image segmentation using k-Means clustering and improved watershed algorithm. *Proceedings of* *the 7th IEEE Southwest Symposium on Image Analysis and Interpretation,* Grand Hyatt Denver Downtown, Denver, Colorado, 26-28 March 2006.

Niuniu, X. & Yuxun, L. 2010. Review of Decision Trees. *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT),* Chengdu, China, 9- 11 July 2010.

Ojala, T. & Pietikainen, M. 1999. Unsupervised texture segmentation using feature distributions. *Pattern recognition, 32*:477-486.

Olaofe, Z. O. & Folly, K. A. 2012. Wind power estimation using recurrent neural network technique. *Proceedings of the IEEE Power Engineering Society Conference and Exposition in Africa (PowerAfrica),* Witwatersrand, Johanneburg, 9-13 July 2012.

Omran, M.G.H. 2004. Particle Swarm optimization methods for pattern recognition and image processing. PhD Thesis, University of Pretoria, South Africa.

Pachghare, V. K., Kulkarni, P. & Nikam, D. M. 2009. Intrusion detection system using self organizing maps. *International Conference on Intelligent Agent & Multi-Agent Systems, Chennai, India, 22-24 July 2009.*

Pal, M. 2002. Factor influencing the accuracy of remote sensing classifications: A comparative study. PhD Thesis, University of Nottingham, England.

Pan, J.S. 1996. Improved algorithms for VQ Codeword search, codebook design and codebook index assignment. PhD Thesis, University of Edinburgh, United Kingdom.

Partio, M., Cramariuc, B., Gabbouj, M. & Visa, A. 2002. Rock texture retrieval using gray level co-occurrence matrix. *Proceedings of the 5th Nordic Signal Processing Symposium board Hurtigruten M/S Trollfjord, Norway, 4-7 October 2002.*

Pass, G. & Zabih, E. 1996. Histogram refinement for content based image retrieval. *Proceedings of the third IEEE workshop on Applications of Computer Vision (WACV'96),* Sarasota, Florida, 2-4 December.

Pass, G., Zabih, R. & Miller, J. 1996. Comparing images using color coherence vectors. http://www.ossipovorchestra.ru/images/Comparing%20Images%20Using%20Color%20Coherence%20Vectors.pdf. [12 August 2012].

Pathak, B. & Barooah, D. 2013. Texture analysis based on the gray level co-occurrence matrix considering possible orientations. *International Journal of advanced research in Electrical electronics and Instrumentation Engineering,* 2(9):4206-4212.

Patil, P. G. & Hegadi, R. S. 2014. Classification of offline handwritten signatures using wavelets and a pattern recognition neural network. Proceedings of IJCA on National Conference on *Recent advances in Information Technology NCRAIT (4):*5-9, February 2014.

Polaschegg, M. 2005. *Study of a CubeSat mission.* MSc. Thesis, Karl Franzens University of Craz, Austria.

Prasad, S. S., Savitri, T. S. & Krishna, I. M. 2011. Classification of multispectral satellite images using clustering with svm classifier. *International Journal of Computer Applications , 35* (5): 32-44, December.

Pressman, R. S. 2005. *Software Engineering: A practitioner's approach*. 6th Edition. New York: McGrawHilll International Edition.

Purandare, A. 2004. Unsupervised word sense discrimination by clustering similar contexts. MSc. Thesis, University of Minnesota, United States.

Puzicha, J., Held, M., Ketterer, J., Buhmann, J. M. & Fellner, D. W. 2000. On spatial quantization on color images. *IEEE Transactions on image processing, 9*(4): 666-682.

Qian, G., Sural, S. & Pramanik, S. 2002. A comparative analysis of two distance measures in color image databases. *Proceedings of the International Conference on Image Processing (ICIP), Rochester, New York, 22-25 September 2002.*

Raghupathi, G., Anand, R. S. & Dewal, M. L. 2010. Color and textures features for content based image retrieval. *Proceedings of he 2nd International Conference on multimedia and content based image retreval, 21-23 July 2010.*

Rahimi, E., Shokouchi, S. B. & Sadr, A. 2007. A parallelized and Pipedlined Datapath to implement isodata algorithm for rosette scan images on a reconfigurable hardware. *The IEEE International Conference on Granular Computing, Silicon Valley, California, 2-4 November 2007.*

Rai, P. & Singh, S. 2010. A survey of clustering techniques. *International Journal of Computer Applications,* 7(12):1-5, October.

Ramasundaram, S. & Victor, S. P. 2013. Algorithms for text categorization: A comparative study. *World Apllied Sciences Journal , 22*(9):1232-1240.

Ramdas, B. P., Keshav, S. B. & Pradeep, M. P. 2012. Wavelet transform techniques for image resolution enhancement. *International Journal of Emerging Technology and Advanced Engeering, 2:* 62-65, April.

Rao, C. S., Kumar, S. S. & Mohan, B. C. 2010. Content based image retrieval using exact legendre moments and support vector machine. *The International Journal of Multimedia & ITS Applications , 2*(2):69-79, May.

Rehman, M. Z. & Nawi, N. H. 2011. Improving the accuracy of gradient descent back propagation algorithm (GDAM) on classification problems. *International Journal on New Computer Architecture and their Applications (IJNCAA),* 1(4): 838-847.

Rejeshwari, S. & Sharmila, T. S. 2013. Efficient quality analysis of MRI image using preprocessing techniques. *The IEEE Conference on Information and Communication Technology (ICT 2013), JeJu Island, 11-12 April 2013.*

Rongjie, L., Jie, Z., Pingjian, S., Fengjing, S. & Guanfeng, L. 2008. An agglomerative hierarchical clustering based high resolution remote sensing image segmentation algorithm. *Proceedings of the International Conference on Computer Science and Software Engineering (CSSE),* Wuhan, Hubei, China, 12-14 December 2008*.*

Sabale, R. G. & Dani, A. R. 2012. Comparative study of prototype model for software engineering with system development life cycle. *Journal of Engineering, 2*(7):21-24, July.

Salmon, B.P. 2012. Improved hyper-temporal feature extraction methods for land cover change detection in satellite time series. PhD Thesis, University of Pretoria.

Sandau, R. 2003. Potential for high resolution imaging with small satellites. *Proceedings of the IEEE International Conference on Recent Advances in Space Technologies, Istanbul, Turkey, 20-22 November 2003.*

Sandnes, F. E. 2009. Geo-spatial tagging of image collections using temporal camera usage dynamics. *Proceedings of the 10th International symposium on pervasive systems,algorithms and networks, Kaohsiung, Taiwan, 14-16 December 2009.*

Sangita, O. & Dhanamma, J. 2011. An improved k-Means clustering approach for teaching evaluation. In Unnikrishnan, S., Surve, S. & Bhoir, D. (Eds). *Advances in computing, communication and control.* London: Springer: 108-115.

Sayers, C. 1991. Self organizing feature maps and their applications. Technical Report, University of Pennsylvania.

Scholz, A. 2003. Compass-1: Hands on experience for the space engineers of tommorrow. *Proceedings of the IEEE international Conference on Recent Advances in Space Technologies,* Istanbul, Turkey, 20-22 November 2003.

Schor, D., Scowcroft, J. & Kinsner, W. 2009. A command and data handling unit for pico satellite missions. *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCEC), Canada*, 3-6 May 2009.

Serra, J. 1982. Image analysis and mathematical morphology. San Diego: Academic Press.

Seul, M., O'Gorman, L. & Sammon, M. J. 2000. Practical algorithms for image analysis: Description,examples and codes, New York: Cambridge University Press.

Sha, L., Ragunathan, R. & Sathaye, S. S. 1994. Generalized rate-monotonic scheduling theory: A framework for devolping real time systems. *Proceedings of the Institute for Electrical and Electronics Engineers (IEEE), 82* (1):68-82, January.

Shan, C., Gong, S. & McOwan, P. W. 2009. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing, 27*: 803-816.

Sharma, A. K., Sharma, S. S., & Mehta, I. C. 2012. A comparative analysis of software process models. *Proceedings of the National Conference on Recent Trends in Computing NCRTC (5)*, Foundations of Computer Science, New York, 16-20 May 2012.

Sharma, S. & Choudhary, S. 2013. Clinical anatomy image retrieval system using local tera pattern. *International Journal of Advances in Computer Science and Technology, 2*(5):58-64.

Shekar, B. H. & Pilar, B. 2014. Shape representation and classification through pattern spectrum and local binary pattern: A decision level fusion approach. *The 5th International Conference on Signal and Image processing (ICSIP), Bangalore, Karnataka, 8-10 January 2014.*

Shen, W., Wu, G., Sun, Z., Xiong, W., Fu, Z. & Xiao, E. 2011. Study of classification methods of remote sensing image based on Decision Tree technology. *The 2011 International Conference on Computer Science and Service System (CSSS)*, Nanjing, China, 27-29 June 2011.

Shi, Z. H., Li, L., Yin, W., Fang, N. F. & Song, Y.T. 2010. Use of multi-temporal landsat images for analyzng forest transition in relation to socioeconomic factors and the environment. *International Journal of Applied Earth Observation and Geoinformation, 13: 468 - 476.*

Shih, T., Huang, J., Wang, C., Hung, J. & Kao, C. 2001. An Intelligent Content-based Image Retrieval System Based on Color, Shape and Spatial Relations. *Proceedings of the National Science council, Republic of China (ROC), 25(4):232-243.*

Simpson, J. J., McIntire, T.J. & Sienko, M. 2000. An improved hybrid clustering algorithm for natural scenes. *IEEE Transactions on Geoscience Remote Sensing*, 38:1016-1032.

Singh, S. M. & Hemachandran, K. 2012. Content based image retrieval based on the integration of color histogram and gabor texture. *International Journal of Computer Applications,* 59 (17):13-22, December.

Singha, M. & Hemachandran, K. 2012. Content based image retrieval using color and texture. *Signal and Image Processing: An International Journal (SIPIJ) ,3*(1): 39-57, February.

Siraj, F., Salahuddin, M. A. & Yusof, S. A. 2010. Digital image classification for Malaysian blooming flower. *Proceedings of the Second International Conference on Computational Inelligence, Modelling and Simulation,* Bali , Indonesia, 28-30 September 2010.

Sirisha, E., Kumar, S.P., Vishnu, P.H. & Srinivas, Y. 2013. Image retrieval using wavelet decomposition,color correlogram and color mean. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(9):1176-1180, September.

Sivakumar, M. V., Roy, P. S., Harmsen, K. & Saha, S. K. 2003. Satellite remote sensing and GIS application in agricultural meteorology. *Proceedings a training workshop of the World Meteorological Organization,* Dehra Dun, Indian, 11 July 2003.

Sivaram, N. & Ramar, K. 2010. Applicability of clustering and classification algorithms for recruitment data mining. *International Journal of Computer Application,* 4(5): 23-28, July.

Song, M. & Civco, D. 2004. Road extraction using SVM and image segmentation. *Photogrammetric Engineering & Remote Sensing, 70*(12):1365-1371, December.

Sorensen, H. F. 2010. Prognostic classification of overian cancer by LBP texture analysis of microscopy images. M.Tech. Thesis, University of Osloensis, Norway.

Soriano, G. C., & Urano, Y. 2011. Replication with state using the self organizing map neural network. *Proceedings of the 13th International Conference on Advanced Communication Technology ICACT), Gangwon-Do, South Korea, 13-16 February 2011.*

Sparenberg, H., Bruns, V. & Foessel, S. 2013. Uses case optimized data storage of scalable media files. *Proceedings of the 3rd International Conference on Innovative Computing (INTECH), London, 29-31 August 2013.*

Srinivasan, G, N. & Shobha,G. 2008. Statisical texture analysis. *World Academy of Science, Engineering and Technology,* 36, December 2008.

Stricker, M. & Orengo, M. 1995. Similarity of color images. *Proceedings of SPIE 2420: Storage and Retrieval for image and Video Database III, San Jose, 23 March 1995.*

Sugana, N. & Thanushkodi, K. 2010. A novel rough set reduct algorithm for medical domain based on bee colony optimization. *Journal of Computing, 2* (6): 49-54, June.

Sundby, D. 2012. Summarizing image collections. M.Tech. Thesis, University of Tromso, Norway.

Syahrul, E. 2011. Lossless and nearly lossless image compression based on combinatorial transforms. PhD Thesis, University of Bourgogne, France.

Takeda, K., Kise, K. & Iwamura, M. 2011. Real time document image retrieval for a 10 million pages database with a memory efficient and stability improved LLAH. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Beijing, 18-21 September 2011.*

Talib, A., Mahmuddin, M. & Husni, H. 2013. Efficient, compact and dominant color correlograms descriptors for content based image retrieval. *Proceedings of the 5th International conference on Advances in Multimedia (MMEDIA)*, Venice, Italy, 21-26 April 2013.

Tani, J., Nishimoto, J. & Ito, M. 2008. Code development learning between human and humanoid robot using a dynamic neural network model. *IEEE Transactions on system, Man and Cybernetics, Part B: Cybernetics, 38:* 43-59.

Tarabalka, Y., Benediktsson, J.A. & Chanussot, J. 2009. Spectral-spatial classification of hyperspectral imagery based on partitional clustering techniques. *IEEE Transactions on Geoscience and Remote Sensing, 47(8): 2973 – 2987.*

Taranto, C., Di-Mauro, N. & Ferillli, S. 2010. Approximate image color correlograms. *Proceedings of the 17th International Conference on Multimedia, Firenze, Italy, 25 October.*

Teng, F. 2006. *A content based image retrieval system for fish taxonomy.* MSc. Thesis, University of New Orleans, United States.

Thakur, D., Markandaiah, N. & Raj, S. D. 2010. Re-optimization of ID3 and C4.5 Decision Tree. *Proceedings of the International Conference on Computer and communication Technology (ICCCT),* Allahabad, Uttar Pradesh ,17-19 September 2010*.*

Thamburaj, F. & Ganapathy, G. 2010. Analysis of genome signature strength of sars coronavirus using self organizing map neural network. *Proceedings of the International*

Conference on Communication and Computational Intelligence, Kong Engineering College, Perundurai, Erode, India, 27-29 December 2010.

Traussnig, W. 2007. Design of a communication and navigation subsystem for CubeSat mission. MSc.Thesis, Karl Franzens University, Graz, Austria.

Tsui, I. & Karam, O. 2011. *Essentials of software engineering.* London: Jones & Bartlett.

Tungkashthan, A. & Intarasema, S. 2009. Spatial color indexing using ACC algorithm. *Proceedings of the 7th International Conference on ICT and Knowledge Engineering*, Bangkok, Thailand, 2 December.

Turky, A. M. & Ahmad, M. S. 2010. The use of SOM for fingerprint classification. Proceedings of the International Conference on Information Retrieval & Knowledge Management (CAMP), Shah Alam Convention Centre, Selangor, Malaysia, 17-18 March 2010.

University of Toronto Institute of Aerospace Studies (UTIAS). 2013. The Canadian advanced nanospace experinment-1 (CANX-1). http://www.utias-sfl.net/nanosatellites/canx-1 [13 November 2013].

Vacha, P. 2010. Query by pictorial example*.* PhD Thesis, Charles University, Prague.

Vailaya, A., Figueiredo, M.A.T., Jain, A. K. & Zhang, H. J. 2001. Image classification for content based indexing. IEEE Transactions on image processing, 10(1):117-129.

Vailaya, A., Jain, A. & Zhang, H. J. 1998. On Image Classification: City images vs. landscapes. *Pattern Recognition*, 31(12):1921-1935, December.

Velmurugan, T. 2012. Effeciency of k-Means and k-Medoids algorithms for clustering arbitary data points. *International Journal Computer Technology and applications*, 3(5):1758-1764.

Verma, M., Srivastava, M.,Chack, N., Diswar, A.K. & Gupta, N. 2012. A comparative study of various clustering algorithms in data mining. *International Journal of Engineering Research and Applications (IJERA), 2(3):1379-1384, May-June.*

Vogel, J. & Schiele, B. 2007. Semantic modeling of natural scenes for content-based image retrieval. International Journal of Computer Vision 72(2):133-157, April.

Wan, C., Wang, C., & Song, X. 2012. A MapReduce based isodata algorithm. *Proceedings of the 3rd International Conference on Intelligent Control and Information Processing, Dalian, China, 15-17 July 2012.*

Wang, J. 2013. *Spatially enhanced local binary patterns for face detection and recognition in mobile device applications.* MSc. Thesis, University of Toronto.

Wang, Z. J. 2001. *Integrated region-based retrieval.* Massachusetts: Kluwer academic.

Wang, Z., Liu, G. Q. & Guo, J. C. 2009. An improved k-Means algorithm based on multiple feature points. Proceedings of the International Workshop on Intelligent Systems and Applications, Wuhan, China, 23-24 May 2009.

Webb, A. & Copsey, K. 2011. *Statistical pattern recognition. Third Edition,* London: Wiley.

Wei, J. M., Wang, S. Q., Yu, G., Gu, L., Wang, G. Y. & Yuan, X. J. 2009. A novel method for pruning Decision Trees. *Proceedings of the 8th International Conference on Machine Learning and Cybernetics,* Baoding, China,12-15 July 2009.

Wells, N. 2007. An investigation into texture features for image retrieval. BSc. Thesis, University of Bath, United Kingdom.

Westfall, L. 2010. *The certified software quality engineers handbook.* Milwaukee: Quality Press.

Wilson, H. G., Boots, B. & Millward, A. A. 2002. A comparison of hierarchical and partitioning clustering technique for multispectral image classification*. IEEE International Geoscience and Remote Sensing Symposium,* Toronto, Canada, 24-28 June 2002*.*

Witten, I. H. & Frank, E. 2005. *Data Mining: practical machine learning tools and technique.* London: Elsevier.

Wu, L. & Wen, Y. 2009. Weed/corn seedling recognition by support vector machine using texture features. *Africa Journal of Agricultural Research, 4*(9): 840-846,September.

Wu, X. & Kumar. 2009. The top ten algorithms in data mining, Boca Raton: Chapman Hall.

Xianchuan, X. & Qi, Z. 2009. Medical image retrieval using local binary patterns with image Euclidean distance. *The 2009 International Conference on Information Engineering and Computer Science (ICIECS 2009)*, Wuhan, China, 19-20 December 2009.

Xiao-Shuai, X., Qing-Quan, Z., Pei-Lin, Y. & Zhao-Yang, L. C. 2010. Research on BP algorithm based on conjugate gradient. *Proceedings of the 2nd International Conference on Information Science and Engineering (ICISE)*, Hangzhou, China, 4-6 December 2010.

Xinwu, L. 2010. Research on text clustering algorithm based on improved k-Means. *Proceedings of the 2010 International conference on computer design and applications, Northeastern University, Qinhuangdao, China, 25-27 June 2010.*

Xu, B., Yin, Q., & Lv, G. 2009. Using SVM to organize the image database. *Proceedings of the IEEE International Conference on Computational Intelligence and Security,* Beijing, China, 11-14 December 2009.

Xu, R. & Wunsch, D.C. 2009. *Clustering.* New Jersey: Wiley.

Xue, C., Xiaowen, L. & Jianwen, M. A. 2004. Urban change detection based on self organizing feature map neural network. *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS),* Anchorage Alaska, 20-24 September 2004.

Yan, M. 2005. Methods of determining the number of clusters in a data set and new clustering creterian. PhD Thesis,Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Yang, C. C., Prasher, S. O., Landry, J. A., Perret, J. & Ramaswamy, H. S. 2000. Recognition of weeds with image processing and heir use with fuzzy logic for procision farming. *Canadian Agriculural Engineering , 42* (4):195-200.

Yang, J. & Blum, R. S. 2005. *A statistical signal processing approach of image fusion using hidden Markov models .* In Blum, R.S. & Liu, Z. 2005. (Eds). Multi-Sensor Image Fusion and Its Applications. Boca Raton: CRC Press.

Yi, F. & Moon, I. 2013. Extended k-Means algorithm. *Proceedings of the 2013 fifty International Conference on Intelligent Human Machine Systems and Cybernetics, Hangzhou, China, 26-27 August 2013.*

Yong, Y., Chixi, W., Bencheng, Y. & Zhi-Hao, Y. 2013. Research on the method of image preprocessing in licence plate location*. Proceedings of the International Conference of Computational and Information Sciences,* Shiyan, Hubain, China, 21-23 July 2013*.*

Yu, G., Vladimirova, T. & Sweeting, M. N. 2009. Image compression systems on board satellites. *Acta Astronautica, 64*: 988-1005, February.

Yu, Z., Zhao, Y., & Wang, X. F. 2008. Research advances and prospects of mathematical morphology in image processing. Proceedings of the *IEEE Conference on Cybernetics and Intelligent Systems, Chengdu, China, 21-24 September 2008.*

Nzeugaing, G. N. 2013. Image compression system for a 3U CubeSat. MTech Thesis, Cape Peninsula University of Technology, Cape Town.

Zare, M. R., Seng, W. C. & Mueen, A. 2013. Automatic classification of medical X-ray images. *Malaysian Journal of Computer Sciences, 26* (1): 9-22.

Zhang, D. 2002. Image retrieval based on shape. PhD Thesis, Monash University, Melbourne, Australia.

Zhang, D., & Lu, G. 2004. Review of shape representation and description techniques. *Pattern recognition,* 37*:* 1-19.

Zhang, S. C., Fang, B., Liang, Y. Z., Wen, J. & Wu, L. 2011. A face clustering method based on facial shape information. *Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition, Guilin, 10-13 July 2011.*

Zhang, X. Q., Yang, K. & Hao, B. Q. 2010. Cell-edge detection method based on Canny algorithm and mathematical morphology. *Proceeding of the 3rd International Congress on Image and Signal Processing (CISP2010),* Yantai, China, 16-18 October 2010.

Zhao, H., Wang, H. & Khan, M.K. 2011. Steganalysis for palette based images using generalized difference image and color correlogram. *Signal processing*, 91(11): 2595-2605.

Zheng, N. & Xue, J. 2009. *Statistical learning and pattern analysis for image and video processing.* London: Springer .

Zhong, M., Georgiopoulos, M. & Anagnostopoulos, G. 2008. K-Norm pruning algorithm for Decision Tree classifiers based on error rate estimation. *Machine Learning, 71(1):* 55-88.

Zhou, C. Y. & Chen, Y. Q. 2006. Improving nearest neighbor classification with cam weighted distance. *Pattern Recognition , 39*:635-645.

Zhou, S., Zhang, S. & Karypis, G. 2012. (eds). Advanced data mining and applications. *Proceedings of the 8th International Conference,Advanced Data Mining Applications (ADMA) 2012,* Nanjing, China ,December, 2012.

Appendix A

Listing A.1: ColorFeatureGUI.m code

```
function varargout = ColorFeatureGUI(varargin)
% COLORFEATUREGUI M-file for ColorFeatureGUI.fig
%      COLORFEATUREGUI, by itself, creates a new COLORFEATUREGUI or raises
the existing
%      singleton*.
%
%      H = COLORFEATUREGUI returns the handle to a new COLORFEATUREGUI or
the handle to
%      the existing singleton*.
%
%      COLORFEATUREGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in COLORFEATUREGUI.M with the given input
arguments.
%
%      COLORFEATUREGUI('Property','Value',...) creates a new
COLORFEATUREGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ColorFeatureGUI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ColorFeatureGUI_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 ColorFeatureGUI

% Last Modified by GUIDE v2.5 13-Apr-2014 12:06:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ColorFeatureGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @ColorFeatureGUI_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 ColorFeatureGUI is made visible.
function ColorFeatureGUI_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 ColorFeatureGUI (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

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

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 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
% --- 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 button press in pushbutton1.
function MeasureWBtn_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)

% --- Executes on button press in pushbutton2.
function searchBtn_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)

% --- Executes on button press in pushbutton3.
function QueryDatabaseBtn_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- 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
```

```
% --- Executes during object creation, after setting all properties.
function axes8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes8 (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 axes8
% --- Executes during object creation, after setting all properties.
function axes9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes9 (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 axes9

% --- Executes during object creation, after setting all properties.
function axes10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes10 (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 axes10

% --- Executes during object creation, after setting all properties.
function axes11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes11 (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 axes11
```

Listing A.2: *ShapeFeatureGUI.m* code

```
function varargout = ShapeFeatureGUI(varargin)
% SHAPEFEATUREGUI M-file for ShapeFeatureGUI.fig
%      SHAPEFEATUREGUI, by itself, creates a new SHAPEFEATUREGUI or raises
the existing
%      singleton*.
%
%      H = SHAPEFEATUREGUI returns the handle to a new SHAPEFEATUREGUI or
the handle to
%      the existing singleton*.
%
%      SHAPEFEATUREGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in SHAPEFEATUREGUI.M with the given input
arguments.
%
%      SHAPEFEATUREGUI('Property','Value',...) creates a new
SHAPEFEATUREGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ShapeFeatureGUI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ShapeFeatureGUI_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 ShapeFeatureGUI
% Last Modified by GUIDE v2.5 13-Apr-2014 12:10:23
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ShapeFeatureGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @ShapeFeatureGUI_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 ShapeFeatureGUI is made visible.
function ShapeFeatureGUI_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 ShapeFeatureGUI (see VARARGIN)
% Choose default command line output for ShapeFeatureGUI
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes ShapeFeatureGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = ShapeFeatureGUI_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;

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 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


% --- 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 button press in pushbutton1.
function MeasureWBtn_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)
% --- Executes on button press in pushbutton2.
function searchBtn_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)
% --- Executes on button press in pushbutton3.
function QueryDatabaseBtn_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% --- 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


% --- Executes during object creation, after setting all properties.
function axes8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes8 (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 axes8


% --- Executes during object creation, after setting all properties.
```

```
function axes9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes9 (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 axes9


% --- Executes during object creation, after setting all properties.
function axes10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes10 (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 axes10

% --- Executes during object creation, after setting all properties.
function axes11_CreateFcn(hObject, eventdata, handles)
```

Listing A.3: *TextureFeatureGUI.m* code

```
function varargout = TextureFeatureGUI(varargin)
% TEXTUREFEATUREGUI M-file for TextureFeatureGUI.fig
%      TEXTUREFEATUREGUI, by itself, creates a new TEXTUREFEATUREGUI or
raises the existing
%      singleton*.
%
%      H = TEXTUREFEATUREGUI returns the handle to a new TEXTUREFEATUREGUI
or the handle to
%      the existing singleton*.
%
%      TEXTUREFEATUREGUI('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in TEXTUREFEATUREGUI.M with the given input
arguments.
%
%      TEXTUREFEATUREGUI('Property','Value',...) creates a new
TEXTUREFEATUREGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before TextureFeatureGUI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to TextureFeatureGUI_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 TextureFeatureGUI

% Last Modified by GUIDE v2.5 13-Apr-2014 12:15:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TextureFeatureGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @TextureFeatureGUI_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 TextureFeatureGUI is made visible.
function TextureFeatureGUI_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 TextureFeatureGUI (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = TextureFeatureGUI_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;
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 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

% --- 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 button press in pushbutton1.
function MeasureWBtn_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)

% --- Executes on button press in pushbutton2.
function searchBtn_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)


% --- Executes on button press in pushbutton3.
function QueryDatabaseBtn_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- 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


% --- Executes during object creation, after setting all properties.
function axes8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes8 (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 axes8

% --- Executes during object creation, after setting all properties.
function axes9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes9 (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 axes9


% --- Executes during object creation, after setting all properties.
function axes10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes10 (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 axes10

% --- Executes during object creation, after setting all properties.
function axes11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes11 (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 axes11
```

Listing A.4: DataCLassifyGUI.m code

```
function varargout = DateClassifyGUI(varargin)
% DATECLASSIFYGUI M-file for DateClassifyGUI.fig
%      DATECLASSIFYGUI, by itself, creates a new DATECLASSIFYGUI or raises
the existing
%      singleton*.
%
%      H = DATECLASSIFYGUI returns the handle to a new DATECLASSIFYGUI or
the handle to
%      the existing singleton*.
%
%      DATECLASSIFYGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in DATECLASSIFYGUI.M with the given input
arguments.
%
%      DATECLASSIFYGUI('Property','Value',...) creates a new
DATECLASSIFYGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before DateClassifyGUI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to DateClassifyGUI_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 DateClassifyGUI
% Last Modified by GUIDE v2.5 13-Apr-2014 12:24:48
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DateClassifyGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @DateClassifyGUI_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

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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 just before DateClassifyGUI is made visible.
function DateClassifyGUI_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 DateClassifyGUI (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes DateClassifyGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = DateClassifyGUI_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 MeasureWBtn_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)


% --- Executes on button press in pushbutton2.
function searchBtn_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)


% --- Executes on button press in pushbutton3.
function QueryDatabaseBtn_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- 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


% --- Executes during object creation, after setting all properties.
function axes8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes8 (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 axes8


% --- Executes during object creation, after setting all properties.
function axes9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes9 (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 axes9


% --- Executes during object creation, after setting all properties.
function axes10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes10 (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 axes10


% --- Executes during object creation, after setting all properties.
function axes11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

Listing A.5: *IntegratedGUIs.m* Code

```
function varargout = IntegratedGUIs(varargin)
% INTEGRATEDGUIS M-file for IntegratedGUIs.fig
%      INTEGRATEDGUIS, by itself, creates a new INTEGRATEDGUIS or raises
the existing
%      singleton*.
%
%      H = INTEGRATEDGUIS returns the handle to a new INTEGRATEDGUIS or the
handle to
%      the existing singleton*.
%
%      INTEGRATEDGUIS('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in INTEGRATEDGUIS.M with the given input
arguments.
%
%      INTEGRATEDGUIS('Property','Value',...) creates a new INTEGRATEDGUIS
or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before IntegratedGUIs_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to IntegratedGUIs_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 IntegratedGUIs

% Last Modified by GUIDE v2.5 13-Apr-2014 12:47:22

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



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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 just before IntegratedGUIs is made visible.
function IntegratedGUIs_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 IntegratedGUIs (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

```
% --- Outputs from this function are returned to the command line.
function varargout = IntegratedGUIs_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 MeasureWBtn_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)


% --- Executes on button press in pushbutton2.
function searchBtn_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)


% --- Executes on button press in pushbutton3.
function QueryDatabaseBtn_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- 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


% --- Executes during object creation, after setting all properties.
function axes8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes8 (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 axes8


% --- Executes during object creation, after setting all properties.
function axes9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes9 (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 axes9


% --- Executes during object creation, after setting all properties.
function axes10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes10 (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 axes10


% --- Executes during object creation, after setting all properties.
function axes11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes11 (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 axes11
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (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 edit4 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (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 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


% --- 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 button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (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 edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (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 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


% --- 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 edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (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 edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (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 slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject    handle to slider5 (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


% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (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
```

Listing A.6: DateClassify.m Code

```
[FileName, folder] = uigetfile({'*.jpg'}, 'Select File');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
img = imread(fullName);
img = reshape(img,[],1);
A = FileName;
B = folder;
[m,n]=size(A);
for i=1:n
    Value1 = imfinfo(A);
    S(i) = struct('Ti',Value1);
    S(i);
end
for i=1:n
    Fn{i}=getfield(S(1,i).Ti,{1,1},'Filename');
    Dn{i}=getfield(S(1,i).Ti,{1,1},'FileModDate');
    tf = isfield(S(1,i).Ti,'Filename');
    if tf==1
        Fn{i}=getfield(S(1,i).Ti,{1,1},'Filename');
    else Fn{i}=char('data not available');
    end
    main = struct('File',Fn{i},'DatabaseOfImages',Dn{i});
    var = struct2cell(main);
end
Dtime = datenum(Value1.FileModDate);
datestr(Dtime);
[y, m, d, h, mn, s] = datevec(Dtime);
sprintf('Date: %d/%d%d',y,m,d);

%==========================================================
% Store Images of: 2012,2013,2014 (January)
%==========================================================

if (y == 2012 && m == 1)
```

```
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'January2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from January2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 1)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'January2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from January2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2014 && m == 1)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'January2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
```

```
    curs = exec(conn, 'select imagedate from January2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end
%===========================================================
% Store Images of: 2012,2013,2014 (February)
%===========================================================

if (y == 2012 && m == 2)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'February2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from February2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 2)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'February2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from February2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 2)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
```

```
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'February2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from February2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end

%===========================================================
% Store Images of: 2012,2013,2014 (March)
%===========================================================

if (y == 2012 && m == 3)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'March2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from March2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 3)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
```

```
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'March2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from March2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 3)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'March2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from March2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end


%==========================================================
% Store Images of: 2012,2013,2014 (April)
%==========================================================

if (y == 2012 && m == 4)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'April2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from April2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
```

```
    AA = curs.data


elseif (y == 2013 && m == 4)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'April2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from April2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014  && m == 4)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'April2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from April2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end

%============================================================
% Store Images of: 2012,2013,2014 (May)
%============================================================

if (y == 2012 && m == 5)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
```

```
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'May2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from May2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 5)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'May2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from May2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 5)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'May2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from May2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data
```

```
else
    %      statements3
end

%============================================================
% Store Images of: 2012,2013,2014 (June)
%============================================================

if (y == 2012 && m == 6)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'June2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from June2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 6)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'June2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from June2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 6)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
```

```
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'June2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from June2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end


%============================================================
% Store Images of: 2012,2013,2014 (July)
%============================================================

if (y == 2012 && m == 7)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'July2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from July2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 7)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'July2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
```

```
    curs = exec(conn, 'select imagedate from July2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 7)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'July2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from July2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end


%===========================================================
% Store Images of: 2012,2013,2014 (August)
%===========================================================

if (y == 2012 && m == 8)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'August2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from August2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 8)
    conn = database('satelliteimageDBs', 'root', '');
```

```
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'August2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from August2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 8)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'August2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from August2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end

%============================================================
% Store Images of: 2012,2013,2014 (September)
%============================================================

if (y == 2012 && m == 9)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
```

```
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'September2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from September2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 9)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'September2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from September2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 9)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'September2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from September2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end

%===========================================================
% Store Images of: 2012,2013,2014 (October)
```

```
%==============================================================

if (y == 2012 && m == 10)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'October2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from October2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 10)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'October2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from October2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 10)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'October2014',
{'filename';'filedata';'imagedate'},exdata1)
```

```
    %Retrieve
    curs = exec(conn, 'select imagedate from October2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %      statements3
end


%============================================================
% Store Images of: 2012,2013,2014 (Novermber)
%============================================================

if (y == 2012 && m == 11)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'Novermber2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from Novermber2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 11)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'Novermber2013',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from Novermber2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 11)
```

```
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a == 1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'Novermber2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from Novermber2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


else
    %     statements3
end


%============================================================
% Store Images of: 2012,2013,2014 (December)
%============================================================

if (y == 2012 && m == 12)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
%     ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'december2012',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from december2012');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data


elseif (y == 2013 && m == 12)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
```

```
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'december2013',
{'filename';'imageContent';'imageDate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from december2013');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

elseif (y == 2014 && m == 12)
    conn = database('satelliteimageDBs', 'root', '');
    a = isconnection(conn);
    if a==1
        disp ('MySQL Database is connected......');
        sprintf('This image was taken: %d/%d/%d  %d:%d:%d\n',y, m,
d,h,mn,s)
    end
    ping(conn);
    if a == 0;
        disp ('connection not established');
    end
    % Mysql Database storing images
    exdata1 = {FileName,img,datestr(Dtime)};
    fastinsert(conn, 'December2014',
{'filename';'filedata';'imagedate'},exdata1)
    %Retrieve
    curs = exec(conn, 'select imagedate from December2014');
    setdbprefs('DataReturnFormat','cellarray')
    curs = fetch(curs);
    AA = curs.data

else
%     statements3
End
```

Listing A.7: Image pre-processing software tool

```
%------------------------------------------------------------------------
%  Program to enhance and remove noise in an image using
%  Image_pre_processing_Software function
% -----------------------------------------------------------------------
function varargout = Image_pre_processing_Software(varargin)
% IMAGE_PRE_PROCESSING_SOFTWARE M-file for
Image_pre_processing_Software.fig
%      IMAGE_PRE_PROCESSING_SOFTWARE, by itself, creates a new
IMAGE_PRE_PROCESSING_SOFTWARE or raises the existing
%      singleton*.
%
%      H = IMAGE_PRE_PROCESSING_SOFTWARE returns the handle to a new
IMAGE_PRE_PROCESSING_SOFTWARE or the handle to
%      the existing singleton*.
%
%
IMAGE_PRE_PROCESSING_SOFTWARE('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in IMAGE_PRE_PROCESSING_SOFTWARE.M with the
given input arguments.
%
%      IMAGE_PRE_PROCESSING_SOFTWARE('Property','Value',...) creates a new
IMAGE_PRE_PROCESSING_SOFTWARE or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Image_pre_processing_Software_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
Image_pre_processing_Software_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_pre_processing_Software

% Last Modified by GUIDE v2.5 29-Apr-2014 17:24:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @Image_pre_processing_Software_OpeningFcn, ...
    'gui_OutputFcn',  @Image_pre_processing_Software_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_pre_processing_Software is made visible.
function Image_pre_processing_Software_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_pre_processing_Software (see
VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


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


function pushbutton1_Callback(hObject, eventdata, handles)

global J;
[FileName,folder] = uigetfile({'*.jpg'},'Select file');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
J = imread(fullName);
%J = rgb2gray(J);
imshow(J,'Parent',handles.axes1);
set(handles.pushbutton1,'userdata',fullName)
set(handles.pushbutton1,'userdata',FileName)
guidata(hObject,handles);

%h = msgbox(' Please do select a filter to use!!!');
%
% function EnhancedButton_Callback(hObject, eventdata, handles)
% global J;
% J = rgb2gray(J);
% gray = medfilt2(J,[3,3]);
% imshow(gray,'Parent',handles.axes3);


function radiobutton4_Callback(hObject, eventdata, handles)
global J;
J = rgb2gray(J);
```

```
gray = medfilt2(J,[3,3]);
imshow(gray,'Parent',handles.axes3);

function radiobutton5_Callback(hObject, eventdata, handles)

global J;
J = rgb2gray(J);
medianFilter = medfilt2(J,[3,3]);
imshow(medianFilter,'Parent',handles.axes3);

function radiobutton6_Callback(hObject, eventdata, handles)

global J;
J = rgb2gray(J);
WienerFilter = wiener2(J,[5 5]);
imshow(WienerFilter,'Parent',handles.axes3);

function radiobutton7_Callback(hObject, eventdata, handles)

global J;
J = rgb2gray(J);
histequ = histeq(J); % Histogram Equalization
imshow(histequ,'Parent',handles.axes3);

clear all;

% global J;
% J = rgb2gray(J);
% A = get(hObject,'String');
% switch A
%     case 'MedianFilter'
% %          gray = medfilt2(J,[3,3]);
% %          imshow(gray,'Parent',handles.axes3);
% %          %set(handles.text1,'FontSize',10);
%            %J = histeq(J); % Histogram Equalization
% %
% %           J = imnoise(J,'gaussian',0,0.025);
%          K = wiener2(J,[5 5]);
%          imshow(K,'Parent',handles.axes3);
%     case 'AverageFilter'
%
%          % imshow(K,'Parent',handles.axes3);
%     case 'WienerFilter'
%           K = wiener2(J,[5 5]);
%          imshow(K,'Parent',handles.axes3);
%     case 'HistogramEqualization'
%          J = histeq(J); % Histogram Equalization
%          imshow(J,'Parent',handles.axes3);
%
% end
```

Listing A.8: Testing color histogram extractor

```
function varargout = TestExtractImageHistogram(varargin)
% TESTEXTRACTIMAGEHISTOGRAM M-file for TestExtractImageHistogram.fig
%      TESTEXTRACTIMAGEHISTOGRAM, by itself, creates a new
TESTEXTRACTIMAGEHISTOGRAM or raises the existing
%      singleton*.
%
%      H = TESTEXTRACTIMAGEHISTOGRAM returns the handle to a new
TESTEXTRACTIMAGEHISTOGRAM or the handle to
%      the existing singleton*.
%
%      TESTEXTRACTIMAGEHISTOGRAM('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in TESTEXTRACTIMAGEHISTOGRAM.M with the
given input arguments.
%
%      TESTEXTRACTIMAGEHISTOGRAM('Property','Value',...) creates a new
TESTEXTRACTIMAGEHISTOGRAM or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before TestExtractImageHistogram_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to TestExtractImageHistogram_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
TestExtractImageHistogram

% Last Modified by GUIDE v2.5 30-Apr-2014 14:24:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TestExtractImageHistogram_OpeningFcn,
...
                   'gui_OutputFcn',  @TestExtractImageHistogram_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 TestExtractImageHistogram is made visible.
```

```
function TestExtractImageHistogram_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 TestExtractImageHistogram (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


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


function pushbutton1_Callback(hObject, eventdata, handles)
[FileName,folder] = uigetfile({'*.jpg'},'Select file');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
J = imread(fullName);
J = rgb2gray(J);
x_axis = 0:16:255;
y_axis = J(1:end);
hist(y_axis,x_axis);
 freq = histc(y_axis,x_axis);
imshow(J,'Parent',handles.axes1);
%imshow( freq,'Parent',handles.axes2);
set(handles.pushbutton1,'userdata',fullName)
set(handles.pushbutton1,'userdata',FileName)
guidata(hObject,handles);

set(handles.edit1,'String',freq(1));
set(handles.edit2,'String',freq(2));
set(handles.edit3,'String',freq(3));
set(handles.edit4,'String',freq(4));
set(handles.edit5,'String',freq(5));
set(handles.edit6,'String',freq(6));
set(handles.edit7,'String',freq(7));
set(handles.edit8,'String',freq(8));
set(handles.edit9,'String',freq(9));
set(handles.edit10,'String',freq(10));
set(handles.edit11,'String',freq(11));
set(handles.edit12,'String',freq(12));
set(handles.edit13,'String',freq(13));
```

```
set(handles.edit14,'String',freq(14));
set(handles.edit15,'String',freq(15));
set(handles.edit16,'String',freq(16));

function edit1_Callback(hObject, eventdata, handles)


function edit1_CreateFcn(hObject, eventdata, handles)

% 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 edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)

% 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 edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)

% 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 edit4_Callback(hObject, eventdata, handles)

function edit4_CreateFcn(hObject, eventdata, handles)

% 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 edit5_Callback(hObject, eventdata, handles)

function edit5_CreateFcn(hObject, eventdata, handles)

% 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 edit6_Callback(hObject, eventdata, handles)

function edit6_CreateFcn(hObject, eventdata, handles)

% 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 edit7_Callback(hObject, eventdata, handles)

function edit7_CreateFcn(hObject, eventdata, handles)

% 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 edit8_Callback(hObject, eventdata, handles)

function edit8_CreateFcn(hObject, eventdata, handles)

% 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 edit9_Callback(hObject, eventdata, handles)

function edit9_CreateFcn(hObject, eventdata, handles)

% 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 edit10_Callback(hObject, eventdata, handles)


function edit10_CreateFcn(hObject, eventdata, handles)

% 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 edit11_Callback(hObject, eventdata, handles)
```

```
function edit11_CreateFcn(hObject, eventdata, handles)

% 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 edit12_Callback(hObject, eventdata, handles)

function edit12_CreateFcn(hObject, eventdata, handles)

% 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 edit13_Callback(hObject, eventdata, handles)

function edit13_CreateFcn(hObject, eventdata, handles)

% 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 edit14_Callback(hObject, eventdata, handles)


function edit14_CreateFcn(hObject, eventdata, handles)

% 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 edit15_Callback(hObject, eventdata, handles)


function edit15_CreateFcn(hObject, eventdata, handles)

% 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 edit16_Callback(hObject, eventdata, handles)
```

```
function edit16_CreateFcn(hObject, eventdata, handles)

% 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 pushbutton2_Callback(hObject, eventdata, handles)
```

Listing A.9: Testing image metadata extractor (Date and Time)

```
%-------------------------------------------------------------------------
%  Program to extract date and time using function ImageMetadataExtractor
% -------------------------------------------------------------------------
function varargout = ImageMetadataExtractor(varargin)
% IMAGEMETADATAEXTRACTOR M-file for ImageMetadataExtractor.fig
%      IMAGEMETADATAEXTRACTOR, by itself, creates a new
IMAGEMETADATAEXTRACTOR or raises the existing
%      singleton*.
%      H = IMAGEMETADATAEXTRACTOR returns the handle to a new
IMAGEMETADATAEXTRACTOR or the handle to
%      the existing singleton*.
%      IMAGEMETADATAEXTRACTOR('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in IMAGEMETADATAEXTRACTOR.M with the given
input arguments.
%      IMAGEMETADATAEXTRACTOR('Property','Value',...) creates a new
IMAGEMETADATAEXTRACTOR or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ImageMetadataExtractor_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ImageMetadataExtractor_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 ImageMetadataExtractor

% Last Modified by GUIDE v2.5 06-May-2014 11:18:35
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ImageMetadataExtractor_OpeningFcn,
...
                   'gui_OutputFcn',  @ImageMetadataExtractor_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 ImageMetadataExtractor is made visible.
function ImageMetadataExtractor_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 ImageMetadataExtractor (see
VARARGIN)
% Choose default command line output for ImageMetadataExtractor
handles.output = hObject;
% Update handles structure
```

```
guidata(hObject, handles);
% UIWAIT makes ImageMetadataExtractor wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = ImageMetadataExtractor_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;
function pushbutton1_Callback(hObject, eventdata, handles)
[FileName,folder] = uigetfile({'*.jpg'},'Select file');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
J = imread(fullName);
A = FileName;
B = folder;
[m,n]=size(A);
for i=1:n
    Value1 = imfinfo(A);
    S(i) = struct('Ti',Value1);
    S(i);
end
for i=1:n
    Fn{i}=getfield(S(1,i).Ti,{1,1},'Filename');
    Dn{i}=getfield(S(1,i).Ti,{1,1},'FileModDate');
    tf = isfield(S(1,i).Ti,'Filename');
    if tf==1
        Fn{i}=getfield(S(1,i).Ti,{1,1},'Filename');
    else Fn{i}=char('data not available');
    end
    main = struct('File',Fn{i},'DatabaseOfImages',Dn{i});
    var = struct2cell(main);
end
Dtime = datenum(Value1.FileModDate);
datestr(Dtime);
[y, m, d, h, mn, s] = datevec(Dtime);
sprintf('Date: %d/%d%d',y,m,d);

imshow(J,'Parent',handles.axes1);
%imshow( freq,'Parent',handles.axes2);
set(handles.pushbutton1,'userdata',fullName)
set(handles.pushbutton1,'userdata',FileName)
guidata(hObject,handles);

set(handles.edit1,'String',FileName);
set(handles.edit2,'String',y);
set(handles.edit3,'String',m);
set(handles.edit4,'String',d);
set(handles.edit5,'String',h);
set(handles.edit6,'String',mn);
set(handles.edit7,'String',s);
function pushbutton2_Callback(hObject, eventdata, handles)
function edit1_Callback(hObject, eventdata, handles)
function edit1_CreateFcn(hObject, eventdata, handles)
% 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 edit5_Callback(hObject, eventdata, handles)
function edit5_CreateFcn(hObject, eventdata, handles)

% 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 edit6_Callback(hObject, eventdata, handles)
function edit6_CreateFcn(hObject, eventdata, handles)

% 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 edit7_Callback(hObject, eventdata, handles)
function edit7_CreateFcn(hObject, eventdata, handles)

% 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 edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)

% 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 edit3_Callback(hObject, eventdata, handles)
function edit3_CreateFcn(hObject, eventdata, handles)
% 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 edit4_Callback(hObject, eventdata, handles)
function edit4_CreateFcn(hObject, eventdata, handles)
% 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
```

Listing A.10: Extract shape Feature algorithm

```
function varargout = TestExtractShapeMorphologyFeatures(varargin)
% TESTEXTRACTSHAPEMORPHOLOGYFEATURES M-file for
TestExtractShapeMorphologyFeatures.fig
%      TESTEXTRACTSHAPEMORPHOLOGYFEATURES, by itself, creates a new
TESTEXTRACTSHAPEMORPHOLOGYFEATURES or raises the existing
%      singleton*.
%
%      H = TESTEXTRACTSHAPEMORPHOLOGYFEATURES returns the handle to a new
TESTEXTRACTSHAPEMORPHOLOGYFEATURES or the handle to
%      the existing singleton*.
%
%
TESTEXTRACTSHAPEMORPHOLOGYFEATURES('CALLBACK',hObject,eventData,handles,...
) calls the local
%      function named CALLBACK in TESTEXTRACTSHAPEMORPHOLOGYFEATURES.M with
the given input arguments.
%
%      TESTEXTRACTSHAPEMORPHOLOGYFEATURES('Property','Value',...) creates a
new TESTEXTRACTSHAPEMORPHOLOGYFEATURES or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before
TestExtractShapeMorphologyFeatures_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
TestExtractShapeMorphologyFeatures_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
TestExtractShapeMorphologyFeatures

% Last Modified by GUIDE v2.5 30-Apr-2014 14:30:51

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@TestExtractShapeMorphologyFeatures_OpeningFcn, ...
                   'gui_OutputFcn',
@TestExtractShapeMorphologyFeatures_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 TestExtractShapeMorphologyFeatures is made
visible.
function TestExtractShapeMorphologyFeatures_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 TestExtractShapeMorphologyFeatures
(see VARARGIN)

% Choose default command line output for TestExtractShapeMorphologyFeatures
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

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

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

function pushbutton1_Callback(hObject, eventdata, handles)
[FileName,folder] = uigetfile({'*.jpg'},'Select file');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
J = imread(fullName);
J = rgb2gray(J);
binary_image = im2bw(J,0.76);
se = strel('disk',3);
openedBW = imopen(binary_image,se);
bw = bwareaopen(openedBW,1000);
bw = imfill(bw,'holes');
[B,L] = bwboundaries(bw,'noholes');
stats = regionprops(L,'Area','Centroid');
threshold = 0.76;
for k = 1:length(B)
    boundary = B{k};
    delta_sq = diff(boundary).^2;
    perimeter = sum(sqrt(sum(delta_sq,2)));
    area = stats(k).Area;
    Roundness = 4*pi*area/perimeter^2;
end
imshow(J,'Parent',handles.axes1);
%imshow( freq,'Parent',handles.axes2);
set(handles.pushbutton1,'userdata',fullName)
set(handles.pushbutton1,'userdata',FileName)
guidata(hObject,handles);

set(handles.edit1,'String',perimeter);
set(handles.edit2,'String',area);
set(handles.edit3,'String',Roundness);

function edit1_Callback(hObject, eventdata, handles)

function edit1_CreateFcn(hObject, eventdata, handles)
```

```
% 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 edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)

% 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 edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)

% 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 pushbutton2_Callback(hObject, eventdata, handles)
```

Listing A.11: Extract Texture Feature algorithm

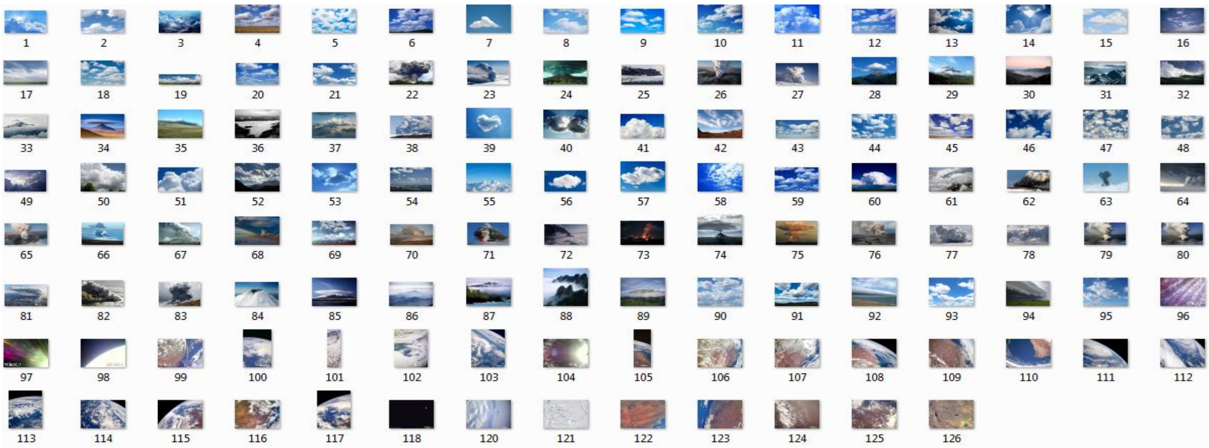
```
function varargout = TestExtractGLCMFeatures(varargin)
% TESTEXTRACTGLCMFEATURES M-file for TestExtractGLCMFeatures.fig
%      TESTEXTRACTGLCMFEATURES, by itself, creates a new
TESTEXTRACTGLCMFEATURES or raises the existing
%      singleton*.
%
%      H = TESTEXTRACTGLCMFEATURES returns the handle to a new
TESTEXTRACTGLCMFEATURES or the handle to
%      the existing singleton*.
%
%      TESTEXTRACTGLCMFEATURES('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in TESTEXTRACTGLCMFEATURES.M with the given
input arguments.
%
%      TESTEXTRACTGLCMFEATURES('Property','Value',...) creates a new
TESTEXTRACTGLCMFEATURES or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before TestExtractGLCMFeatures_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to TestExtractGLCMFeatures_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
TestExtractGLCMFeatures

% Last Modified by GUIDE v2.5 30-Apr-2014 14:01:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TestExtractGLCMFeatures_OpeningFcn,
...
                   'gui_OutputFcn',  @TestExtractGLCMFeatures_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 TestExtractGLCMFeatures is made visible.
function TestExtractGLCMFeatures_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 TestExtractGLCMFeatures (see
VARARGIN)
% Choose default command line output for TestExtractGLCMFeatures
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TestExtractGLCMFeatures wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
 % --- Outputs from this function are returned to the command line.
function varargout = TestExtractGLCMFeatures_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;
function pushbutton1_Callback(hObject, eventdata, handles)
[FileName,folder] = uigetfile({'*.jpg'},'Select file');
if FileName ~= 0
    fullName = fullfile(folder,FileName);
end
J = imread(fullName);
J = rgb2gray(J);
 GLCMS = graycomatrix(J,'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 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 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;
```

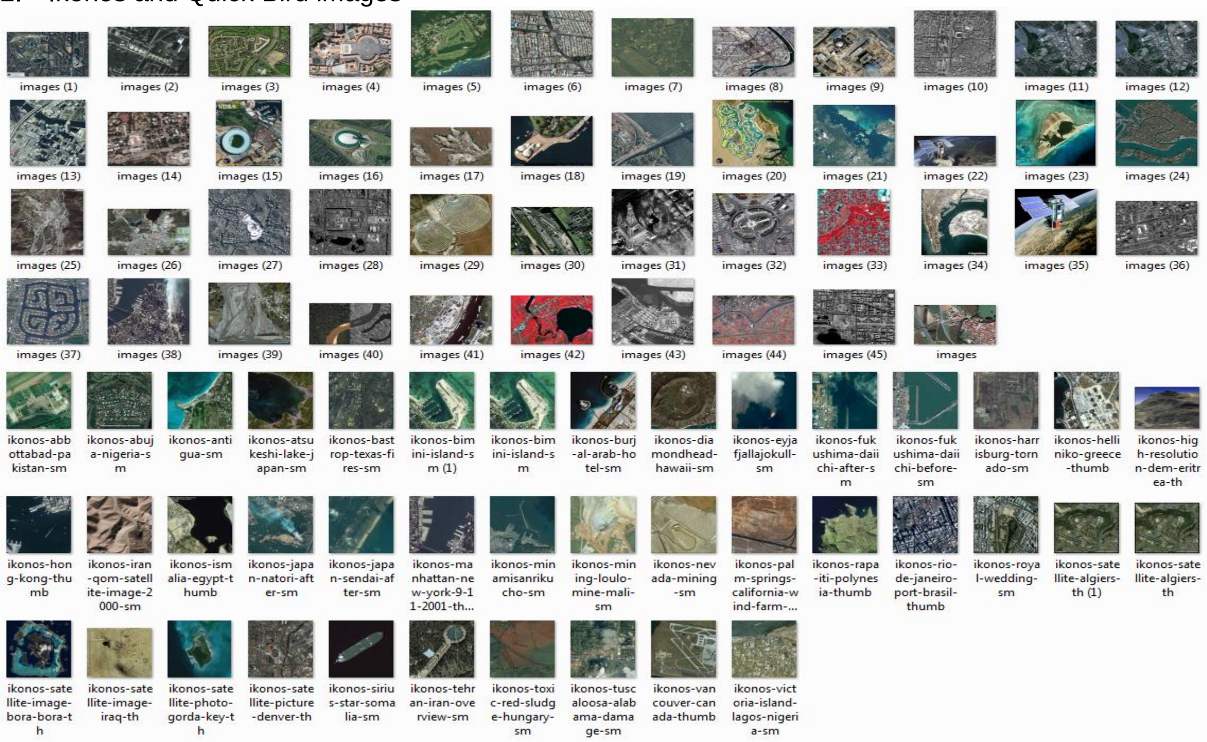
```
end
mean_homogeneity;

imshow(J,'Parent',handles.axes1);
%imshow( freq,'Parent',handles.axes2);
set(handles.pushbutton1,'userdata',fullName)
set(handles.pushbutton1,'userdata',FileName)
guidata(hObject,handles);
set(handles.edit1,'String',mean_contrast);
set(handles.edit2,'String',mean_correlation);
set(handles.edit3,'String',mean_energy);
set(handles.edit4,'String',mean_homogeneity);

function edit1_Callback(hObject, eventdata, handles)

function edit1_CreateFcn(hObject, eventdata, handles)

% 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 edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)

% 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 edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)

% 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 edit4_Callback(hObject, eventdata, handles)

function edit4_CreateFcn(hObject, eventdata, handles)

% 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 pushbutton2_Callback(hObject, eventdata, handles)
```

Appendix B : Excerpt of images used for testing and evaluation

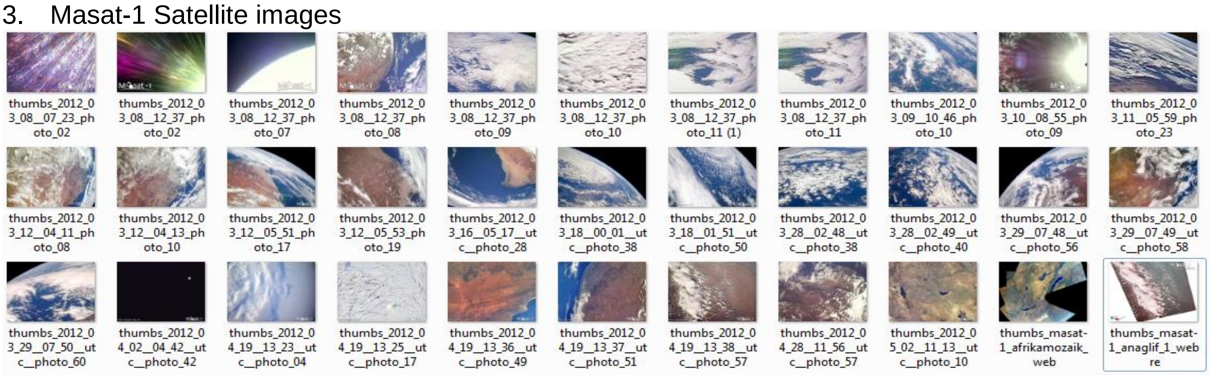
1. Google High Resolution Images



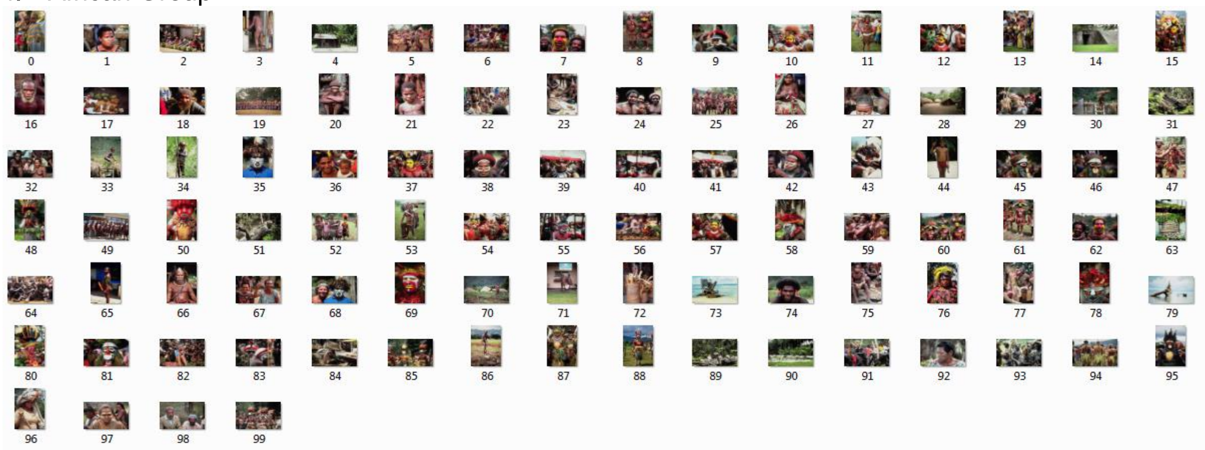
2. Ikonos and Quick Bird images



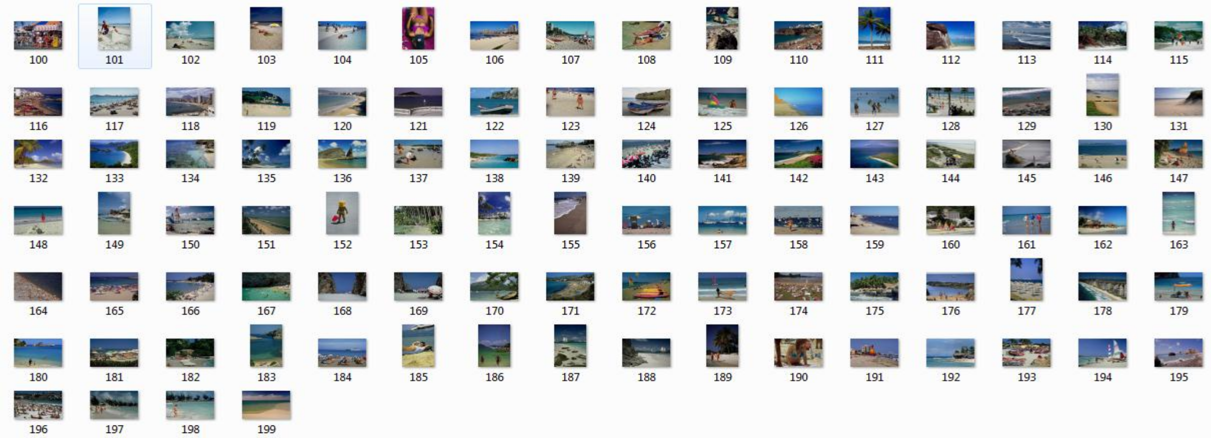
3. Masat-1 Satellite images



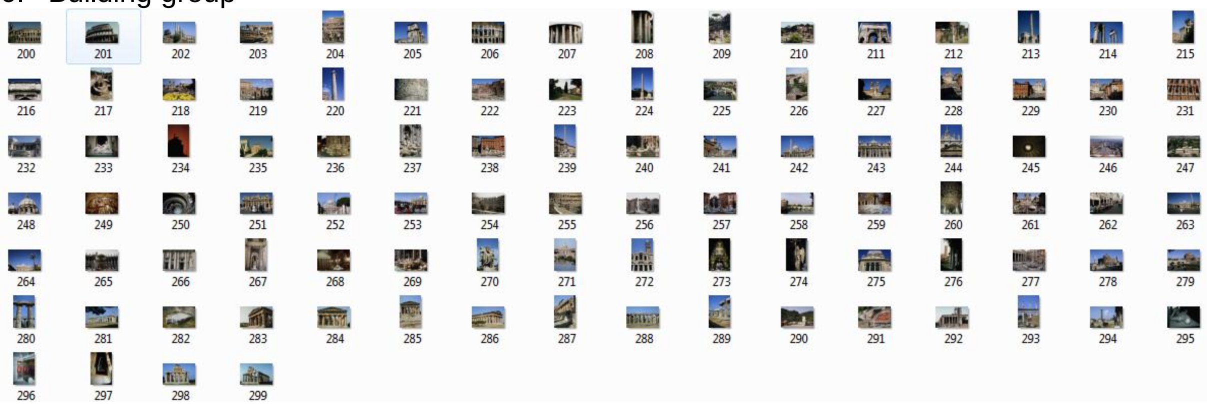
4. African Group



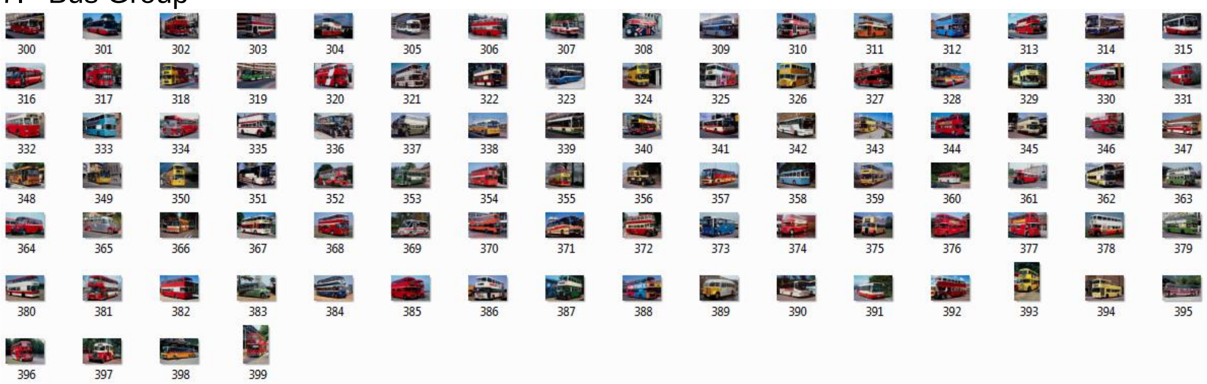
5. Beach group



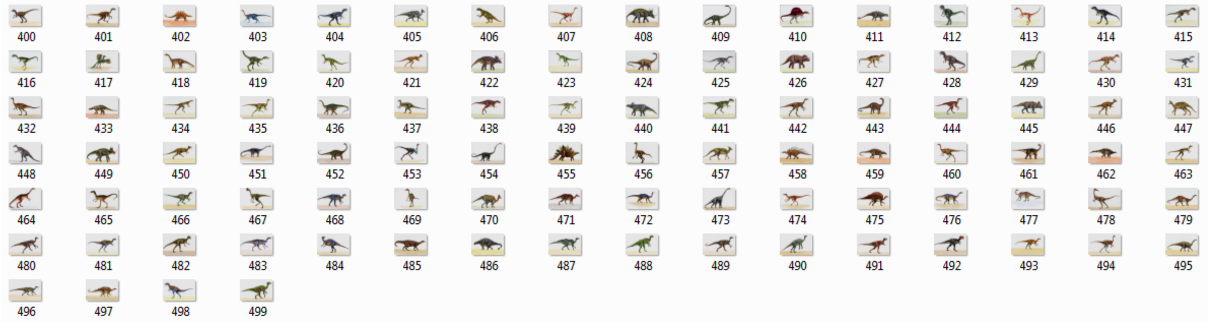
6. Building group



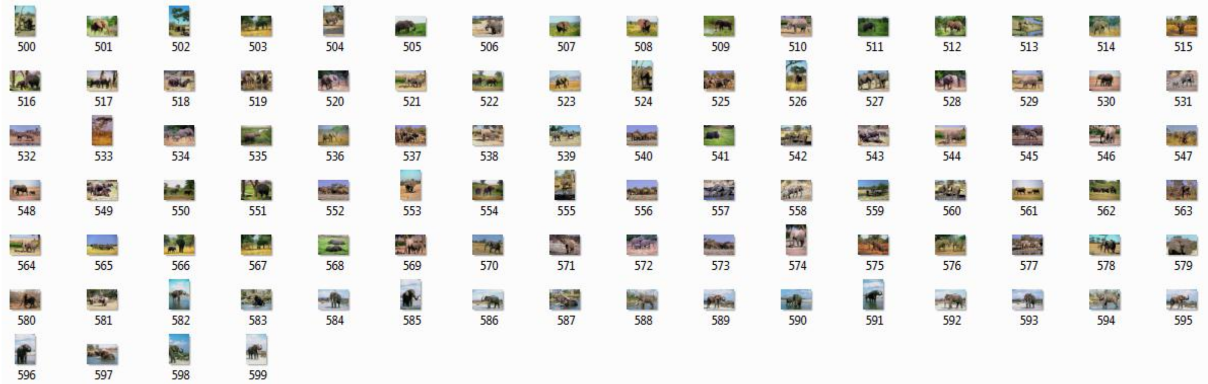
7. Bus Group



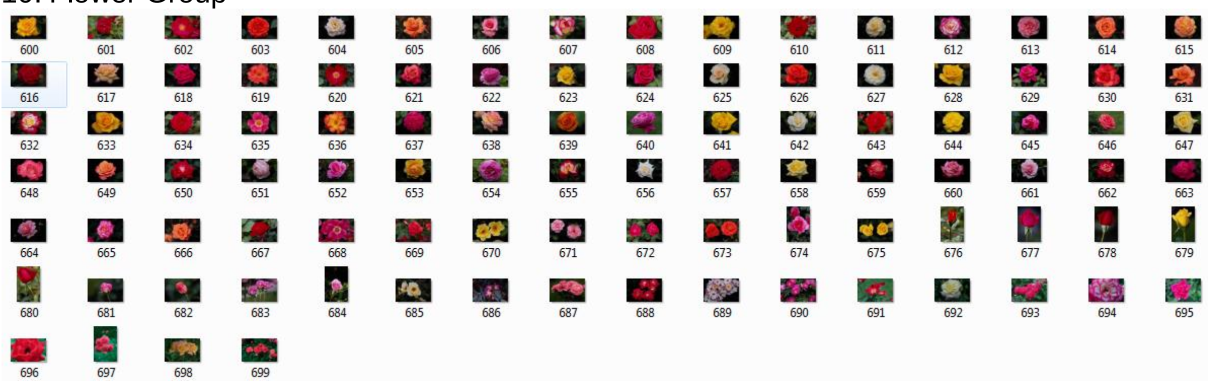
8. Dinosaur group



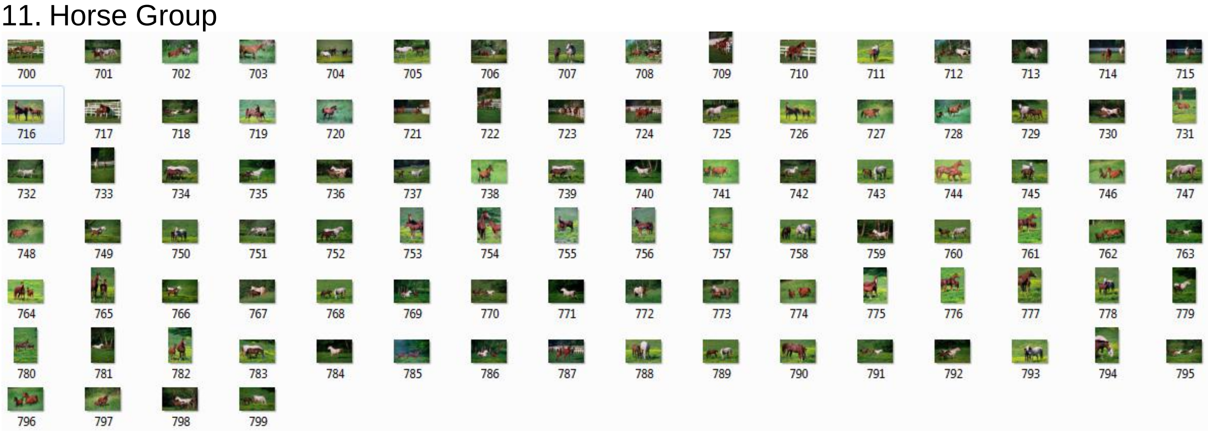
9. Elephant Group



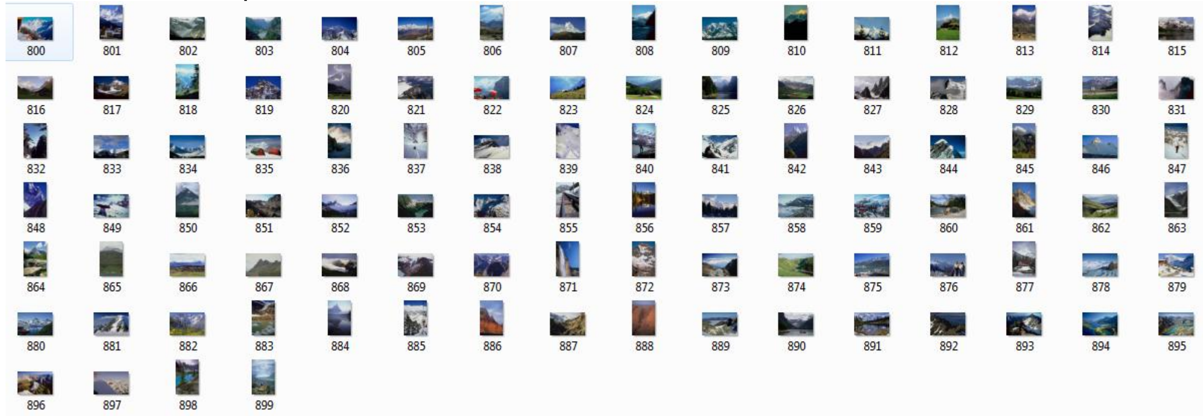
10. Flower Group



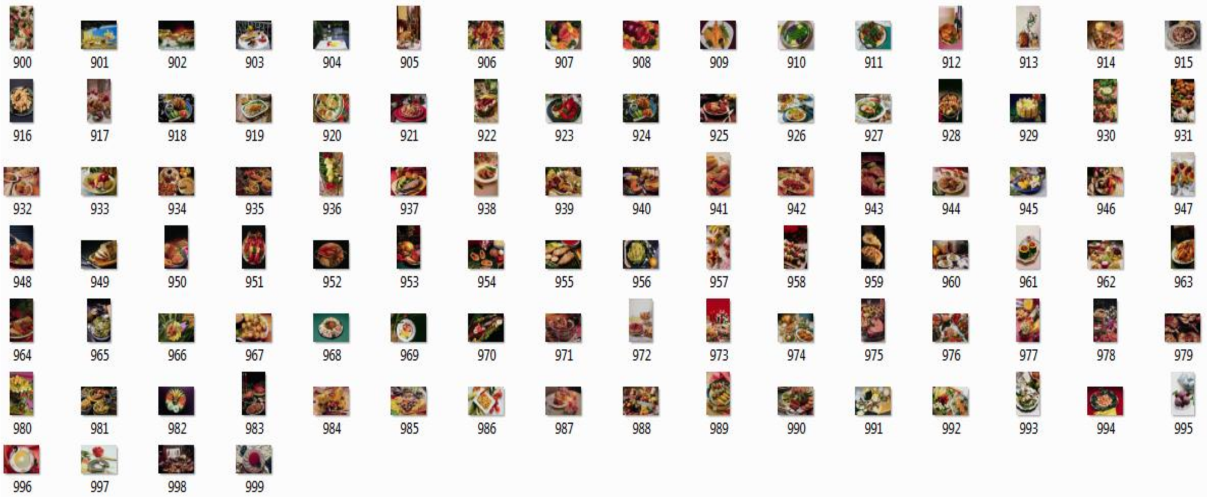
11. Horse Group



12. Mountain Group



13. Food Group



Appendix C: Interfacing Matlab with MYSQL

In order to interface Matlab with MYSQL, the Microsoft Open Database Connectivity (ODBC) needs to be configured (Figure C.1).

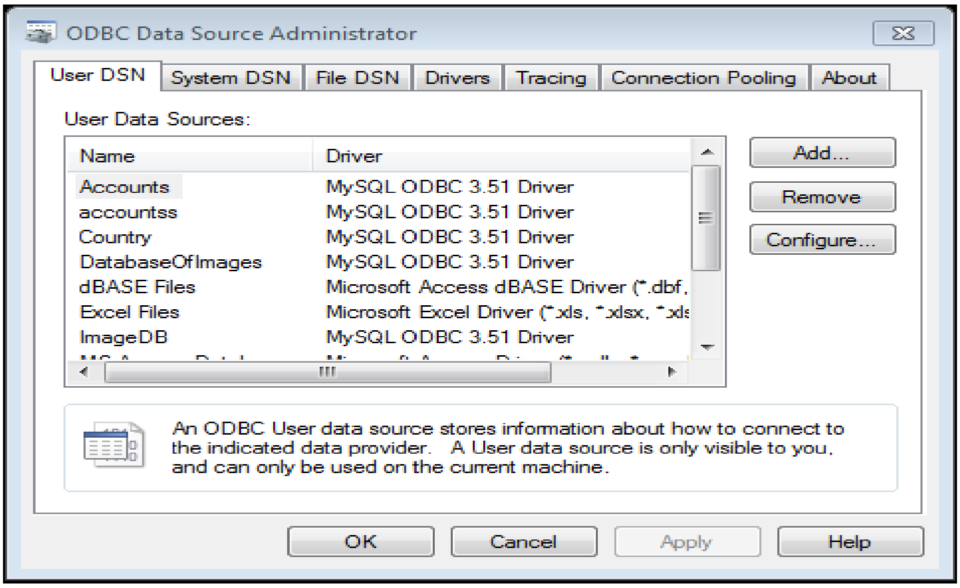


Figure C. 1: ODBC

MYSQL ODBC Driver 3.51 (Figure C.2)

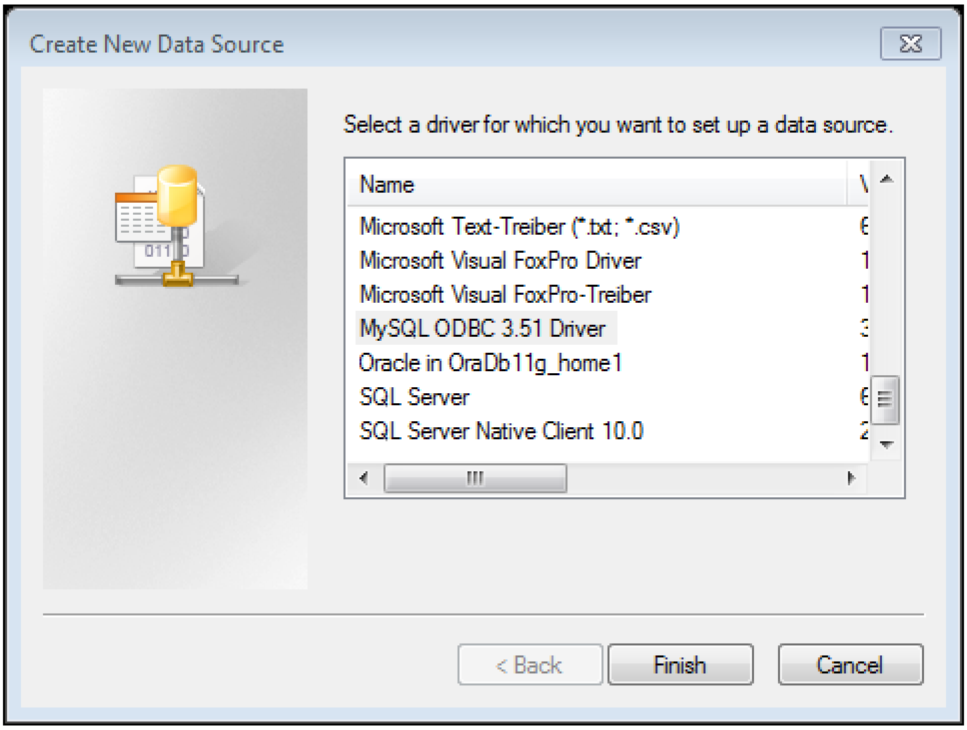


Figure C. 2: MYSQL ODBC Driver

Once a desired driver is selected, fields such as data sourcename, descripition,TCP/IP server, port number, username and password (similar to the one used to download MySQL database) and selection of database to be used to populate are required.

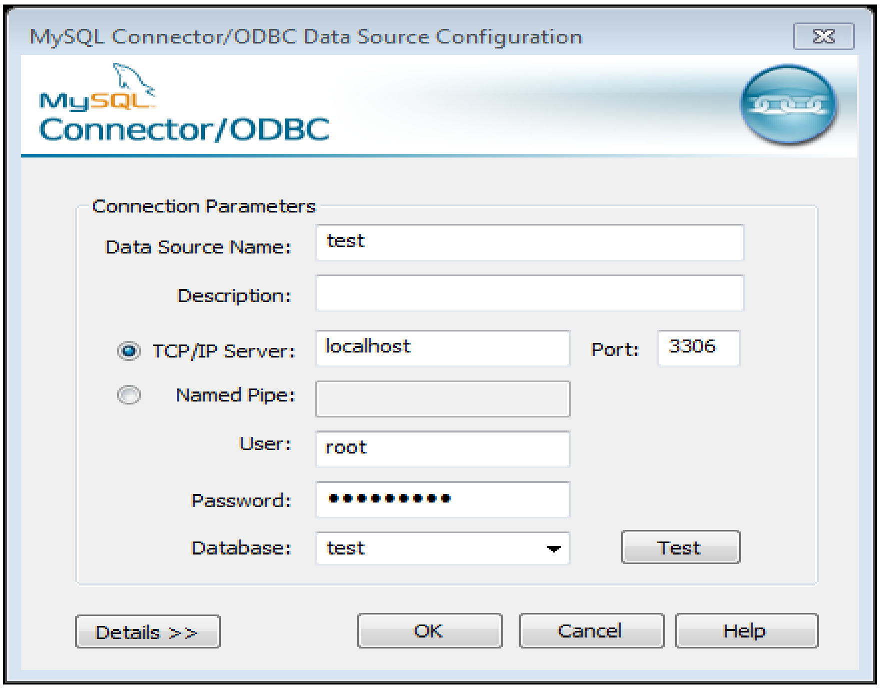


Figure C. 3: Test Window

Once details are filled and testing concluded (Figure C.3) a valid connection is in place (Figure C.4).

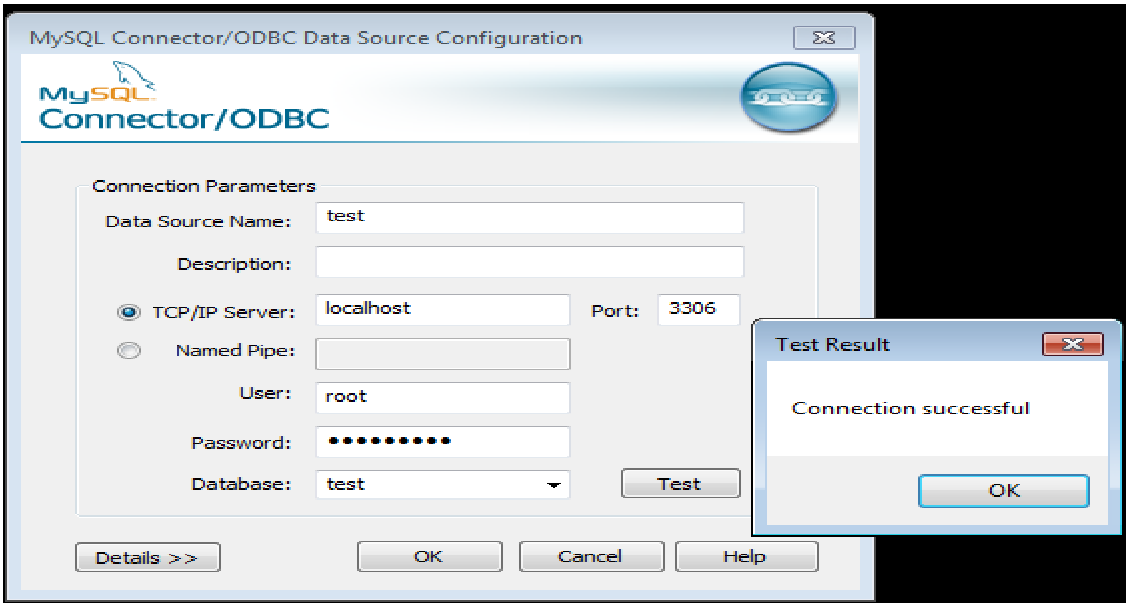


Figure C. 4: Connection status

Once a successful connection is established, the ODBC data source administrator database and the corresponding driver are added (Figure C.5).

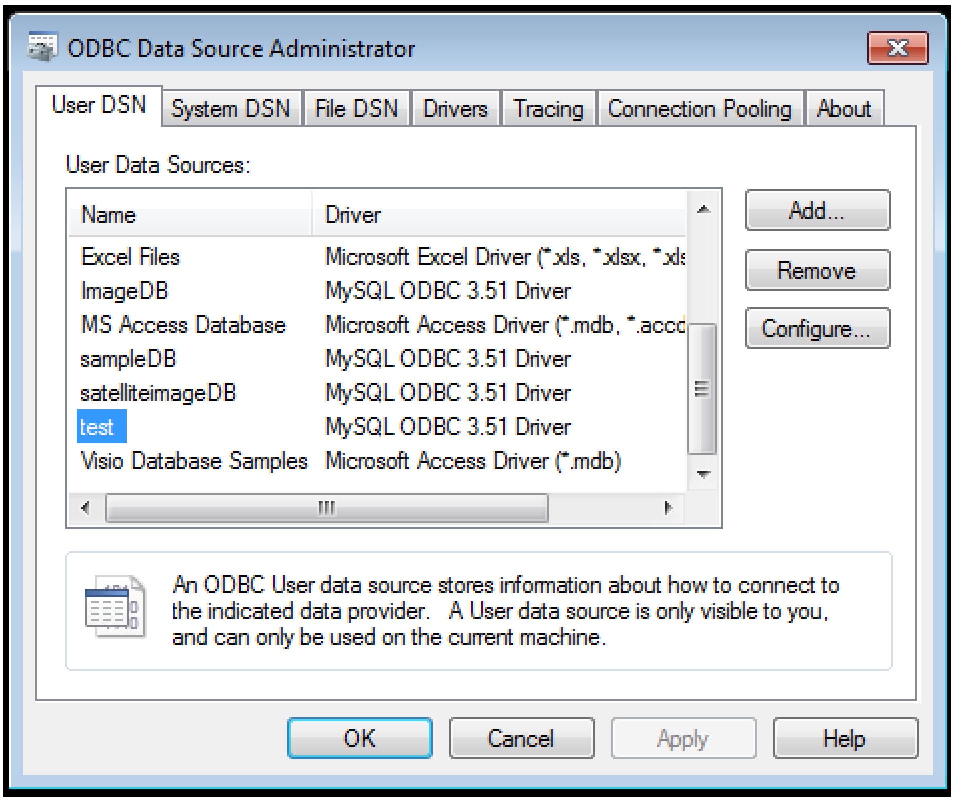


Figure C. 5: ODBC Data Source Administrator

To check whether the MYSQL connection is synchronized with MATLAB, the *logintimeout()* function is used (Figure C.6).

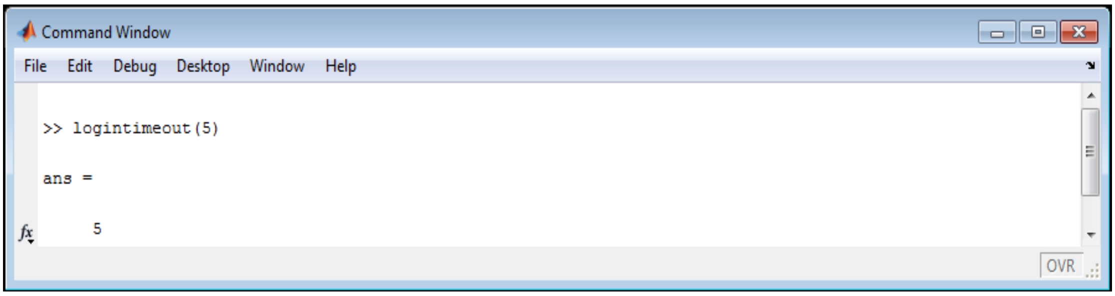


Figure C. 6: MATLAB connection

Figure C.7 and Figure C.8 shows an active connection using the ping command.

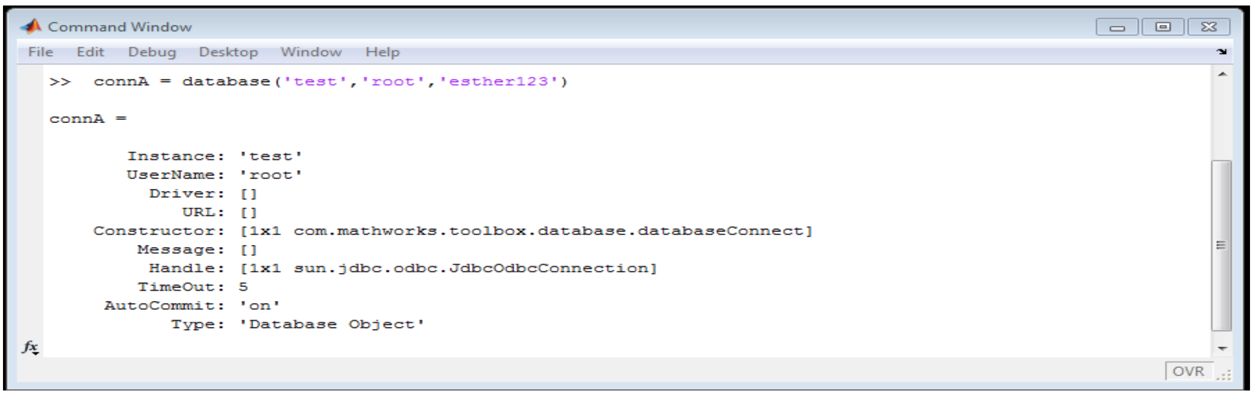


Figure C. 7: Active connection

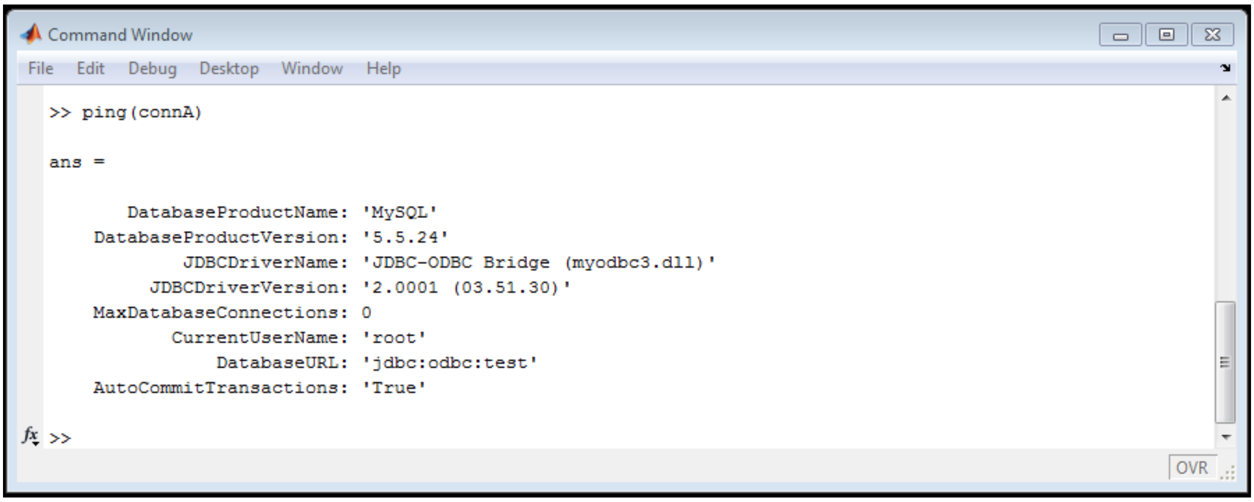


Figure C. 8: Ping

In Figure C.9 a cursor is created to store an SQL command. The fetched keyword is used to limit the rows and always close the connection after use (see listing 1: code).

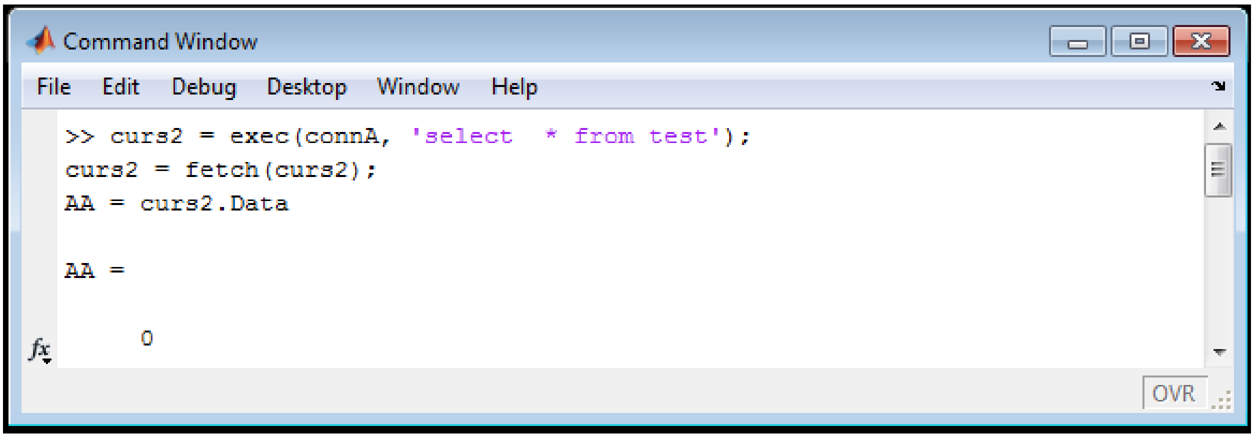


Figure C. 9: Cursor

```
cursorA = exec(connA,'SQL syntax');
cursorA = fetch(cursorA,nRows);
AA = cursorA.data
Example for our project:
curs2 = exec(connA, 'select  * from test');
curs2 = fetch(curs2,10);
AA = curs2.Data

Close your connection after use:
close(cursorA)
close(connA)
```

Listing C.1: Code

Appendix D: Install and Setup of MySQL 5.5 server

Installing MYSQL commence from downloading the application files to a GUI-guided installation wizard. For completeness sake and to guide further projects these steps are highlighted within the next number of figures. The installation is MYSQL 5.5.24 for WIN32.

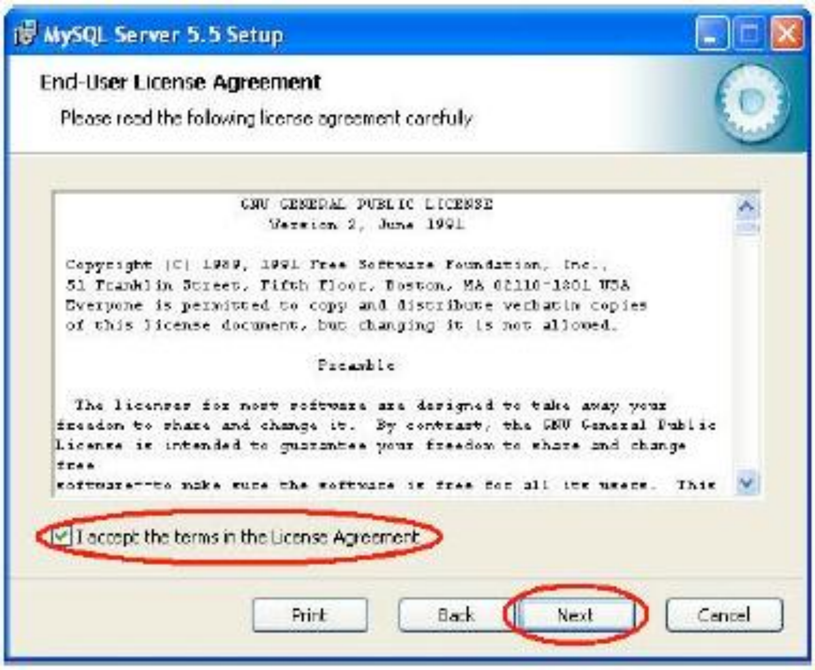


Figure D. 1: License Agreement

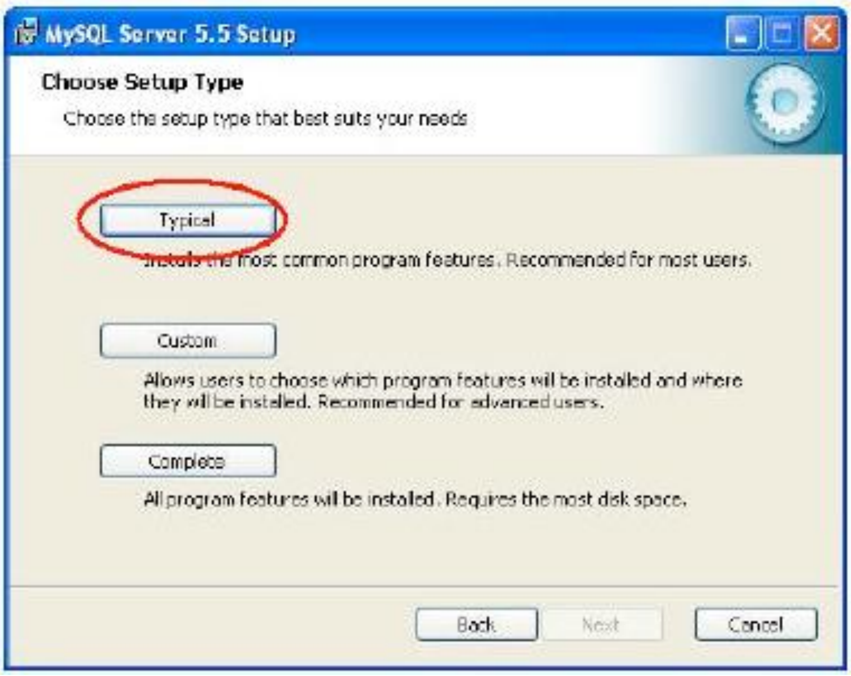


Figure D. 2: Typical Installation

Figure D. 3: Subscription Services

Figure D. 4: Monitoring Services

Figure D. 5: Launch configuration

Figure D. 6: Configuration Wizard

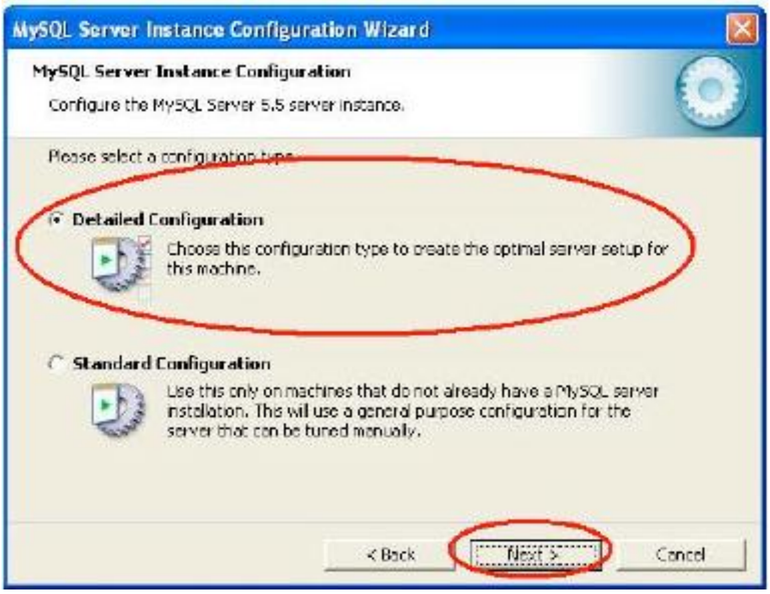


Figure D. 7: Detailed Configuration

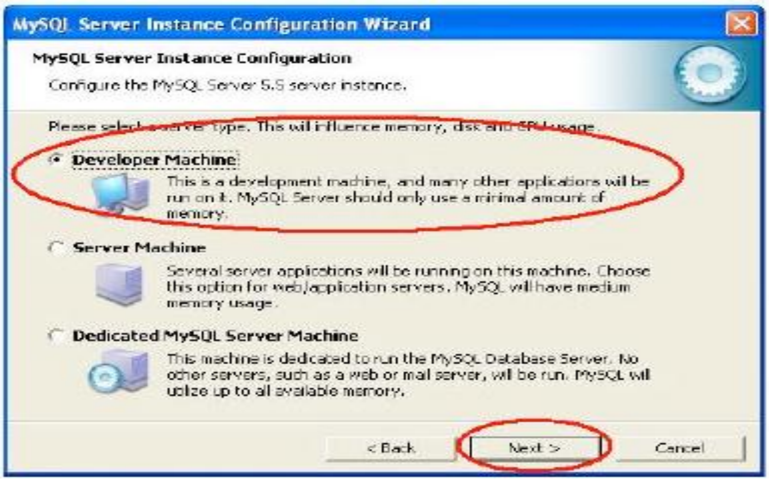


Figure D. 8: Instance Configuration

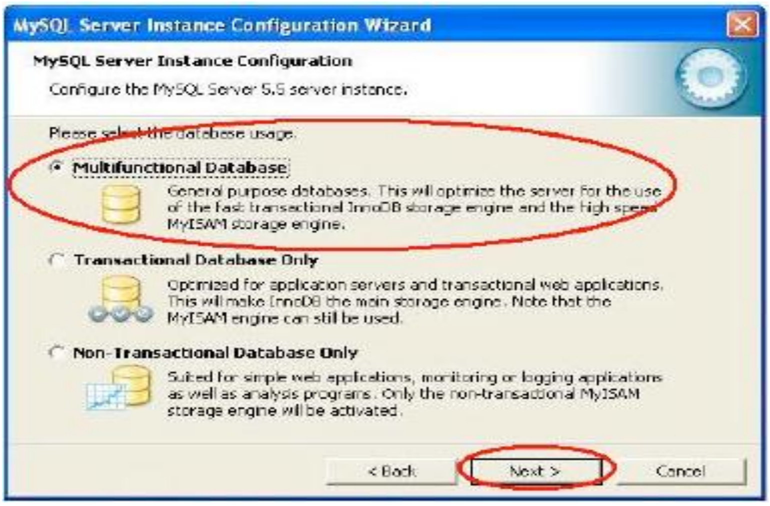


Figure D. 9: Server Instance

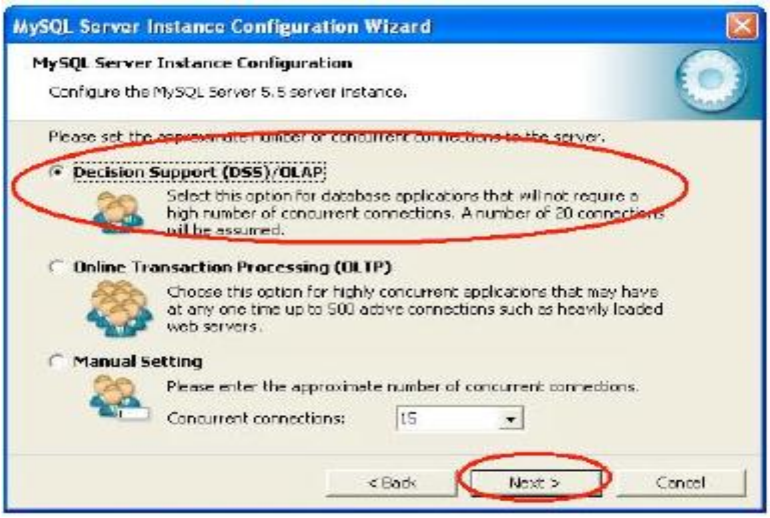


Figure D. 10: Server Instance Configuration

Enable TCP/IP networking and strict mode for SQL mode. Select either default port number (3306) or change it to the desired one.

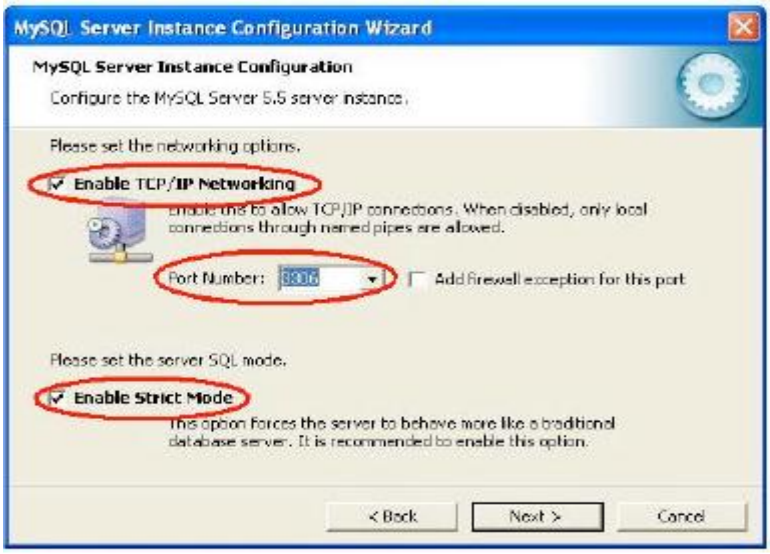


Figure D. 11: TCP/IP and Port

In MySQL server instance configuration wizard window, choose a standard character set.

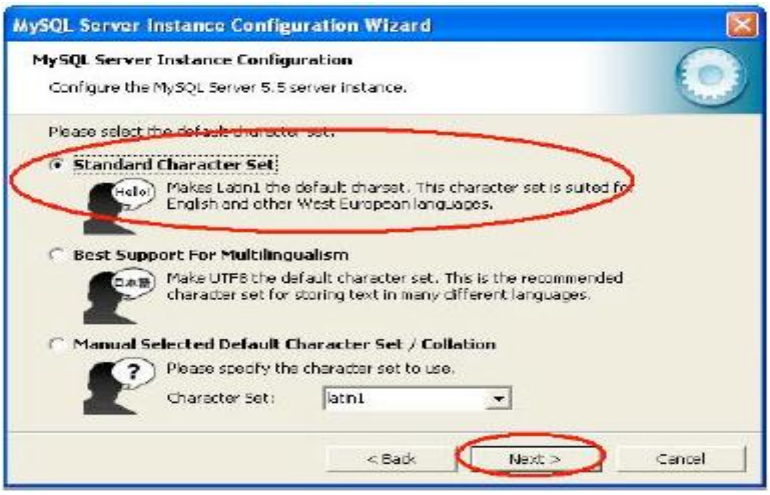


Figure D. 12: Standard Character Set

In the following window below select Install as Window Service, Launch MYSQL server automatically and Include bin directory in Windows PATH.

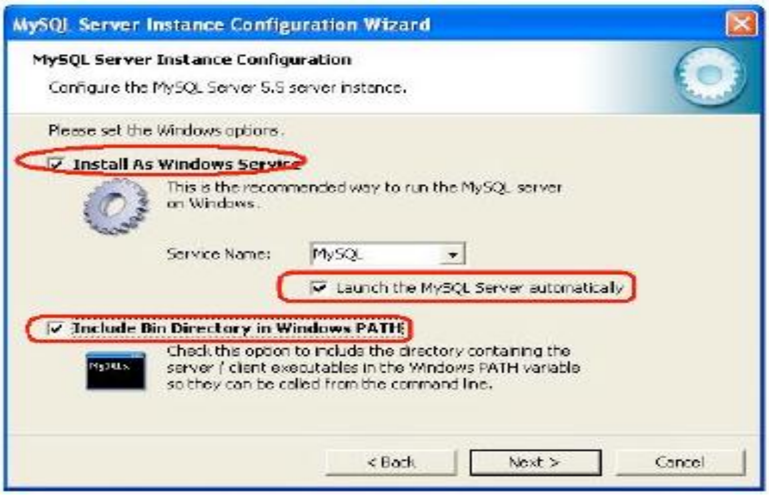
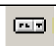
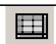
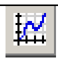
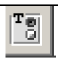
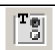


Figure D. 13: Windows Service

Figure D. 14: Authentication

Appendix E: The MATLAB GUIDE Library Components

Table E.1: Guide Library component

Component	Icon	Description
Push button		They cause an action when clicked. Hence, when it's clicked it shows depressed and released appears as raised.
Slider		Allows input of numeric in a specified range and permit the user to shift a slider bar. The positions of the slider specify the relative position within the given range.
Radio button		They are usually, mutually exclusive within a group of related radio buttons and are similar to check boxes. When one radio buttons is selected, other radio buttons in group are deselected.
Check box		Causes an action when checked and indicates their state as checked or not checked. Typically, used when an independent choices are needed.
Edit text		These are fields which lets users to enter or modify text strings. Used when input is needed i.e. numerical input but must be converted to their numeric equivalents.
Static text		Manages display lines of text. Used to label other controls, gives directions to the users or specify values related within a slider and are fixed once set.
Pop-up menu		Provides a list of choices when clicked on.
List box		Display a list of items and allows users to select one or more items.
Toggle button		Causes an action and indicate whether they are turned on or off. Usually a button group is used to manage mutually exclusive toggle buttons.
Table		The table button is used to create a table component. A syntax called uitable is used to create a table in matlab.
Axes		Allows user interfaces to display graphics i.e. images and graphs. Contains properties which can be set to manage numerous aspects of its behaviors and appearance.
Panel		It organizes graphical user interface components into groups. It improves look and feel of user interface and make easier to understand. Panel can have panel children which can be axes, group button and other panels.
Button group		This are similar to panels but are used to control exclusive choice behaviors for radio buttons and toggle buttons.
Toolbar		Help to create toolbars containing push buttons, toggle buttons, save and print icons.
active component		These are available only for Microsoft windows platform, they permits displaying activeX controls in the graphical user interface. They can be child of a figure only, but not a child of panel or button group.