THE UNIVERSITY OF CALGARY

LARGE IMAGERY HANDLING IN RELATIONAL DATABASES FOR GEOSPATIAL INFORMATION SYSTEMS

by

MAN (ROMANNA) GUO

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF GEOMATICS ENGINEERING

CALGARY, ALBERTA

AUGUST, 1999

ABSTRACT

In the last several years, there has been a dramatic increase in the need to incorporate large imagery in relational databases (RDBs) into Geospatial Information System (GISs) to replace traditional GIS database management systems. Large images, such as remotely sensed images including satellite images and airborne images, scanned images and other digital images, are important components of GIS. Although the relational database is known for its maturity and success in handling ordinary thematic information, it is not good at handling large imagery, especially remotely sensed images, due to their large sizes, very rich semantics, and complicated modeling and processing requirements. Some of these impediments can be stated as follows: lack of effective data management utilities; lack of effective indexing methods; lack of data processing abilities and limited capability for powerful query methods.

In order to alleviate these impediments, a prototype of a Large Image Database is proposed in this research which has the following abilities:

1) to effectively and efficiently manage large imagery and its metadata in a single database;
2) to improve performance by using advanced indexing, clustering and partitioning techniques;
3) to manage the database without knowing any SQL syntax;
4) to add easily new functions into the function library to improve database performance;
5) to perform database browsing, query navigation, similarity and content based queries and retrievals.

ACKNOWLEDGEMENTS

I wish to express my deep gratitude to my supervisor, Dr. J.A.R. Blais, for his advice, guidance, discussions and suggestions and also for his encouragement and constant availability throughout my graduate studies. My appreciation also goes to Dr. C. Tao for providing valuable reading material and discussions in remote sensing, image processing, GIS and computer vision.

I am also grateful to Rakhit Petroleum Consulting Ltd. and to Mr. Kaush Rakhit and Mr. Bruce Palmer for allowing me to continue my thesis writing while I am working in Rakhit Petroleum Consulting Ltd.

I also wish to thank the graduate students, Dean Provins and Anthony Warren for their proofreading of an early draft of the thesis, Steve Adam for his providing of testing images and interesting discussions on various topics. I must also thank the secretaries, technicians and computer specialists of the Department of Geomatics Engineering who, in one way or other, helped me to carry out this research.

Finally, my appreciation and thanks go to my dearest husband, Michael. This marks the end to an extremely long educational process for me through which he has faithfully stood beside me, encouraging me to reach my potential and lovingly supporting each stage. Without his continual love, devotion and support, this research could not have been completed. And also my thanks go to my parents and my younger brother for their continual support and encouragement, their endless love and understanding.

Dedication

To my husband, my parents, and my younger brother.

For their continuous support and encouragement.

"Together we stand"

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 Research Background

A Geographic or Geospatial Information System (GIS) is a system of computer hardware, software and procedures designed to support the capture, management, analysis and display of spatially referenced data which are used to solve complex planning and management problems. What type of database management mechanism is used to model and to manipulate spatial and related information is the key contributing factor to maximize system operating efficiency and model extensibility in a GIS implementation.

Traditional GISs, such as ARC/INFO employ two file management systems to manage different kinds of data. One file system, called ARC, manages points, lines and polygons while the other, called INFO, manages the attributes of these spatial objects. They have different user interfaces and different query capabilities. Most GISs have poor or no ability to handle complicated large objects, such as remotely sensed images. Even though there have been dramatic developments in computer and networking techniques over last several years, traditional GIS systems have not kept up with industry demands. Many users are requesting a single repository that can handle all types of data with the same query interface and which has the properties of being distributed, having multi-user access, portable performance, data integrity, back-up and recovery, and has security access control.

The Relational Database Management System (RDBMS) is the most popular and mature database management system and is used by many enterprises. It has requisite capabilities of

distribution, data integrity, back-up and recovery, data query and retrieval, and security access control. Obviously, it can be considered for being employed as a GIS database management system by many database researchers and vendors. Representative and successful implementations include Oracle Spatial Cartridge (SC) [Oracle, 1998] and ESRI Spatial Database Engine (SDE) [ESRI, 1998].

Large imagery is one example of large objects. These objects have come from remotely sensed imagery, scanned maps, digitized video clips, DTMs, one or more dimensional measurements, simulation model output and others [Zhou, 1995]. Each one of these large objects has its own special management methods and processing techniques because each has different data structure and characteristics from the others. This research focuses on large imagery handling in relational databases. This includes remotely sensed images, scanned images and other digital images. It is crucially important to realize that all are important components of GISs. Most are complicated objects that have large data volumes, intensive processing requirements and rich semantics. Commercial RDBMSs are well known for their maturity and success in handling ordinary thematic information, but, they are not good at handling complicated objects such as images and graphics. In fields which need to use large imagery, the benefits of integrating large imagery and its processing techniques into RDBMSs are recognized. They can be summarized as [Zhou, 1995]:

1) Large organizations can establish a single, coherent and integrated corporate information system in a distributed environment.
2) Users can have a unified way to manage and use their information.

3) Users benefit from the perception that they have access to a seamless information environment, uncomplicated by the need to consider differences in data sources, information types, storage devices and processing techniques.
4) By storing all data types in one database, data integrity, transaction, security and recovery can be managed using one database management system (DBMS).
5) Information updating and data quality control can be made easier and more reliable.
6) Decision making can be based on the broader base of information, and is able to address issues that were previously beyond individual data resources.

In addition to the benefits described above, the image data acquisition, reformatting and conversion are usually tedious, time consuming and error prone. Therefore, RDBMSs with powerful GIS functionality which allows users easily to input and to convert large imagery data, which allows them to manage and process complicated imagery data are greatly needed. Some RDBMS researchers and vendors have proposed systems to handle image and graphical data, such as Oracle Image Cartridge [Oracle, 1998]; but they still lack the functionality for handling remotely sensed imagery, or scanned imagery that are important components of GISs.

Other than the industry requirements described above, another motivation of this research are two projects related to large image handling. This research is part of a GIS project for the Crown of the Continent environmental information system, which is one part of the Crown of the Continent Ecosystem Data Atlas (CCEDA). The Crown of the Continent is a unique region of the Western Cordillera between the northern boundary of Yellowstone National

Park and the southern boundary of Banff National Park. The Atlas is designed to be a computerized repository of up-to-date biophysical and socioeconomic research materials covering southwestern Alberta, southeastern B.C. and western Montana. The plans include an integrated data and information system accessible over the Internet with databases distributed among the Calgary and Montana University campuses as well as National Park and other government agency sites in Canada and the U.S. To adequately support environmental research and decision-making related to the Crown, a comprehensive environmental information system is being designed with Internet World Wide Web interactive access and warehousing facilities for distributing integrated data sets to researchers and the interested public [*Blais et al. 1997*].

Also, this research is supported by another project, an IMAGE database project of Geological Survey of Canada (GSC). The IMAGE is a database module and application system intended to manage a wide variety of digital images collected by scientists. The images include satellite imagery, scanned images, DTMs and etc. They vary in scale from microscopic digital SEM images to large-scale satellite images.

Obviously, these two projects involve multidisciplinary users working with lots of large images for environmental applications. Based on the benefits of this research described above, the two projects will greatly benefit from this research. However, there is one thing that should be stressed here that since this research focuses on powerful and sophisticated techniques for large image management and query, the large image databases with numerous images will significantly benefit from this research. However small image databases which

only have tens of images may not require such a complicated management and query system. Therefore the CCEDA project and the IMAGE database of GSC will be the good candidates of employing the proposed system because both of them will deal with hundreds and thousands of images and the sizes of these databases grow quickly.

1.2 Characteristics of Large Images

When compared with thematic data or vector data, large imagery has some special characteristics that are listed below:

1) Thematic data can be well represented with basic and popular data types such as integer, float, string, etc. while the content of a large image cannot be so precisely described. A large image can be described as several groups of classification information or a group of spatial entities (e.g., points, lines, polygons, etc.) with spatial relationships (e.g. adjacency, orientation, relative position, etc.) among them. In general, there are no fixed and/or precise ways to represent these objects and their relationships. Therefore, the data model for large imagery needs to hide the underlying data types for objects and their relationships from users in order to provide mechanisms to map and select the proper representation dynamically.
2) In a large image, spatial entities and relationships do not carry any semantics by themselves. Semantics need to be assigned to entities according to application context. However this will cause some problems with large imagery:

    a) A large image, especially a remotely sensed image, can be interpreted in different

ways by different applications.

b) The same object can show different characteristics when using different sensors or spectral bands during different time periods.

c) Different interpretative results may be generated because of using different techniques.

Therefore, it is important for a database to provide mechanisms to support application-specific processing functions. These can be used to implement user defined object semantics and can overload any system pre-defined methods to perform application-specific tasks. It should also support the building of complex features from simple features [Zhou, 1995]

3) The query methods for large imagery are different from thematic data and other art photo pictures. Due to three reasons described above in item **2)**, the usual text query methods and query methods based on color histograms, which are popularly used for art photo pictures are not good enough. Large complicated images are often queried through other objects. For example, users may want to retrieve an image which is similar to the image displayed on the screen. Depending on the similarity measure used, this may cause multiple or imprecise query results. Therefore, an image database should provide fuzzy query capabilities that allow users to incrementally refine their results using their domain knowledge. Including user defined rules and knowledge in the data model will help in this regard.

4) Large imagery usually involves huge amounts of data. This often results in slow

processing and response time in databases. Therefore, a large imagery database requires capabilities for data indexing, clustering and filtering. To store huge amounts of data in a database also calls for effective data compression and structuring techniques.

5) Metadata and attribute data are often important components of large images, especially in data warehouse and data distribution applications. How to effectively model, manage and derive these metadata needs to be carefully considered in large imagery support.

6) Remotely sensed images sometimes have atmospheric and geometric distortions or are occluded by clouds and noise. Therefore, they require that preprocessing be performed to correct them.

7) Spatial-temporal operations are often needed to allow applications to explore the spatial-temporal behavior of the images.

1.3 Relational Database Model

Since the first database model appeared in the 60s, the evolution has been from flat-file model, through the hierarchic model, the network model, the relational model and now has reached the object-oriented model. Though the object-oriented database model has more capabilities than the relational model for handling large objects and multimedia objects because of its advanced data abstraction techniques, rich semantics support and extensibility, it is a newcomer to the database family and it is far from mature and practical. Relational databases are still and will be the dominant database choice in the marketplace today and for the next several years.

1.3.1 Relational Database Model Definition and Components

The relational database model was developed in 1970 by Dr. E. F. Codd of IBM's San Jose Research Laboratories. At present, there are well over 200 such products commercially available. What is a relational database model? Briefly, it is a system in which:

- The data are perceived by the user as tables (and nothing but tables); and
- The operators at the user's disposal (e.g., for data retrieval) are those that generate new tables from old. For example, there will be one operator to extract a subset of the rows of a given table, and another to extract a subset of the columns -- and of course a row subset and column subset of a table can both in turn be regarded as tables themselves [Date, 1995]

Essentially, there are three basic components of the relational model: relational data structures, constraints that govern the organization of the data structures, and operations that are performed on the data structures.

Relational Data Structures

The relational model supports a single, "logical" structure called a relation. It is a two-dimensional data structure commonly called a table in the "physical" database. Attributes represent the atomic data elements that are related by the relation. For example, Figure 1-1 shows the Customer table. The Customer relation might contain such attributes about a customer as the customer number, customer name, region, credit status, and so on.

The actual data values for the attributes of a relation are stored in tuples, or rows, of the table. It is not necessary for a relation to have rows in order to be a relation; even if no data exists for the relation, the relation remains defined with its set of attributes.

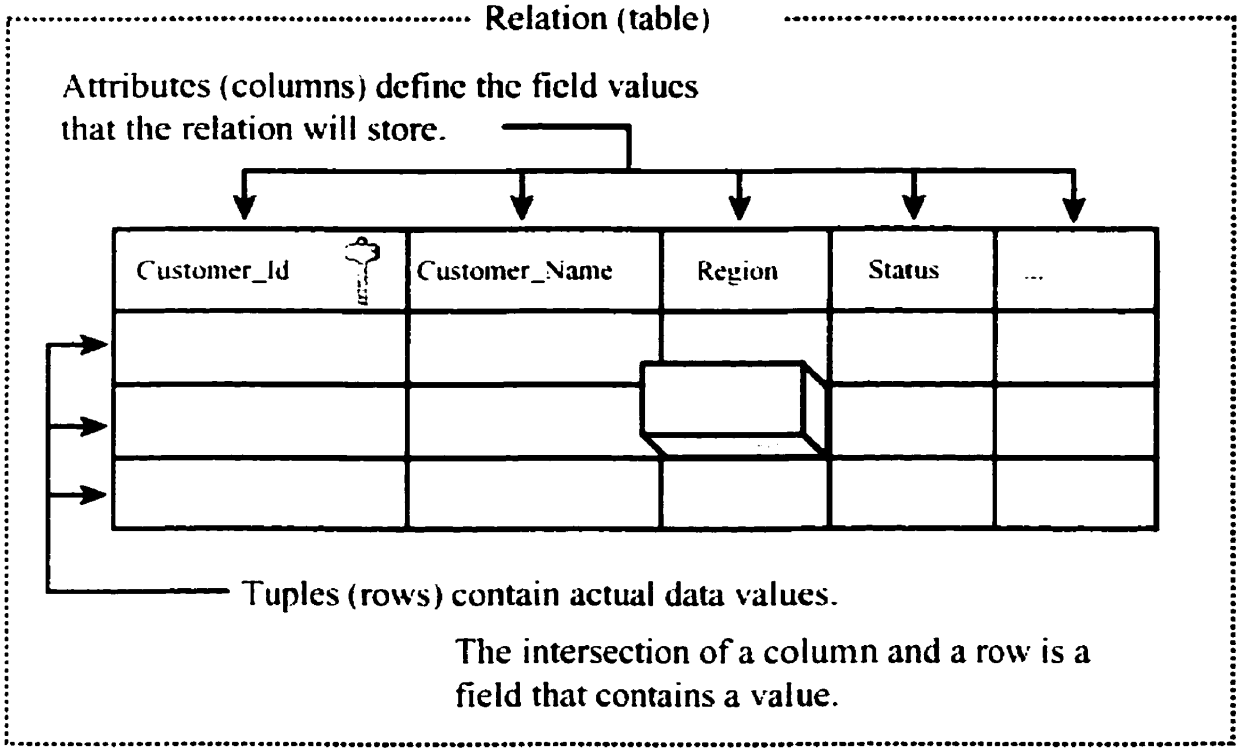


Figure 1-1 Basic Elements of a Relation (adapted from Oracle [1996])

Key Values and Referential Integrity

Attributes are grouped with other attributes based on their dependency on a primary key value. A primary key is an attribute or group of attributes that uniquely identifies a row in a table. A table has only one primary key, and as a rule, every table has one (e.g. in Figure 1-2). Because primary key values are used as identifiers, they cannot be null.

You can have additional attributes in a relation with values that you define as unique to the relation. Unlike primary keys, unique keys can contain null values. In practice, unique keys are used to prevent duplication in the table rather than identify rows. Consider a relation that contains the attribute, United States Social Security Number (SSN). In some rows, this attribute may be null in since not every person has a SSN; however for a row that contains a non-null value for the SSN attribute, the value must be unique to the relation.

Linking one relation to another typically involves an attribute that is common to both relations. The common attributes are usually a primary key from one table and a foreign key from the other. Referential integrity rules dictate that foreign key values in one relation reference the primary key values in another relation. Foreign keys might also reference the primary key of the same relation.

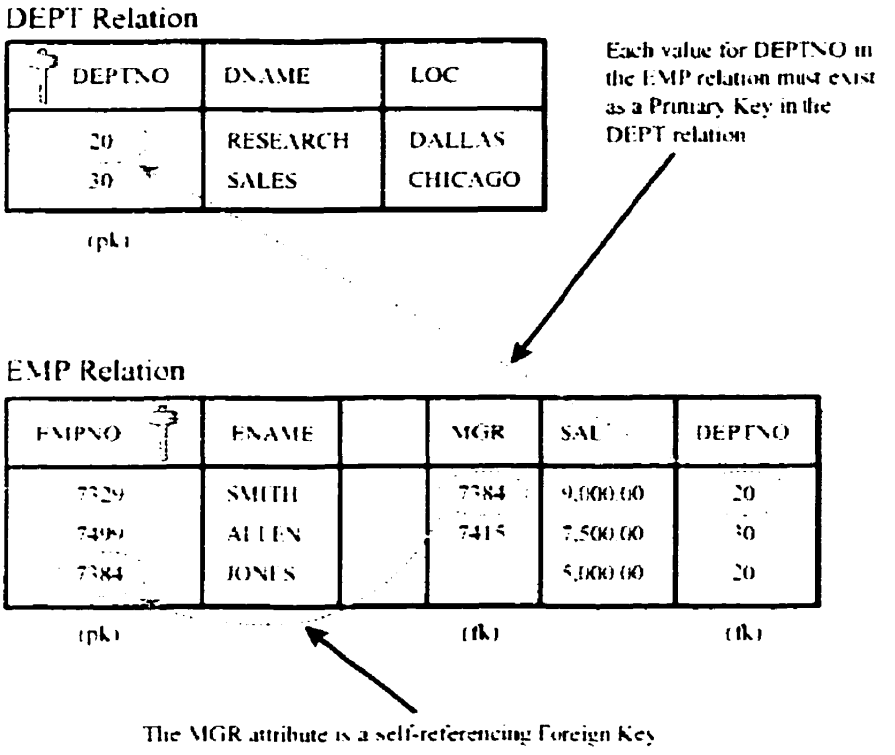


Figure 1-2 Two Foreign Key Relationships (adapted from Oracle [1996])

Relational Algebra

The relational model defines the operations that are permitted on a relation or group of relations. There are unary and binary relational operators, each of which result in another relation. These operations are intuitive and very similar to those used with set operations. The fundamental intent of the algebra is to allow the writing of expressions. These expressions in turn are intended to serve a variety of purposes, including defining the scope of a retrieval, defining the scope of an update, defining security rules and integrity rules, etc. Table 1-1 describes the seven operators used to manipulate relational structures [Oracle Unleashed,

1996]. Binary operator types indicate that the operation uses two relations as operands; unary operators require a single relation as an operand.

Operation	*Type*	*Resulting Relation*
Union	Binary	Rows from the two relations are combined, eliminating duplicate rows
Intersection	Binary	Rows common to two relations
Difference	Binary	Rows that exist in the first relation but not in the second
Projection	Unary	Rows that contain some of the columns from the source relation
Selection	Unary	Rows from the source relation that meet query criteria.
Product	Binary	Concatenation of every row in one relation with every row in another.
Join	Binary	Concatenation of rows from one relation and related rows from another.

Table 1-1 Algebra Operations of the Relational Model

1.3.2 Advantages of a Relational Data Model

Compared to the hierarchic and network models, the relational database model has the following advantages [Brathwaite, 1991].

1) Simplicity -- The end user is presented with a simple data model. His or her requests are formulated in terms of the information content and do not reflect any complexities due to system-oriented aspects. A relational database model is what the user sees, but it is not necessarily what will be implemented physically.
2) Nonprocedural Requests -- Because there is no positional dependency between the relations, requests do not have to reflect any preferred structure and therefore can be nonprocedural.
3) Data Independence -- This should be one of the major objectives of any DBMS. The

relational database model removes the details of storage structure and access strategy from the user interface. The model provides a relatively higher degree of data independence than do the other two models. To be able to make use of this property of the relational data model, however, the design of the relations must be complete and accurate.

1.3.3 Object-Relational Database Model

Although the relational model is much better than either the hierarchic and network models in many aspects, it began to show its inadequacies when the need to store and manipulate complex data in relational databases became apparent. Complex data are a component not only of multimedia applications for the Web, but also in specialized application domains such as medical care (including X-rays, MRI imaging, and EKG traces); geographical, space, and exploration systems (such as maps, seismic data, and satellite images); and even financial systems (such as time-series data). In order to support these complicated objects, relational databases are required to have more data types than those supported by the DBMSs. It should be possible to search, access, and manipulate complex data types in the database with standard SQL, without breaking the rules of the relational data model.

These requirements imply making the relational DBMS more object oriented and have resulted in the development of Object-Relational Database (ORDB) and Object-Relational Database Management Systems (ORDBMSs). ORDBMSs add new object storage capabilities to the relational systems at the core of modern information management systems. These new facilities integrate management of traditional field data, complex objects such as

time series and geospatial data, as well as diverse binary media such as audio, video, images, and applets. By encapsulating methods with data structures, an ORDBMS server can execute complex analytical and data manipulation operations to search and transform multimedia and other complex objects. With the ORDBMS, large imagery and multimedia object support is both possible and effective.

Three of the leading DBMS vendors -- IBM, Informix, and Oracle -- have extended their conventional relational DBMSs to become object/relational DBMSs; however they are still not good enough to support large images, especially remotely sensed imagery. Further discussion of ORDBMS will be in the sequel.

1.4 Major Issues for Large Imagery Support in RDBs

In the relational database model, there are many important issues that have to be dealt with for large imagery support, such as data model issues regarding the modeling of non-primitive, complex objects in a flat table structure, and system performance issues about how to effectively and efficiently handle large imagery, and, to improve query speed, etc. Since this research work focuses only on large imagery management and query aspects, these issues will be discussed in the following sections.

1.4.1 Large Imagery Management Issues

In order to handle large objects in RDBs, most relational DBMS vendors have added the capability to manipulate binary large objects (BLOBs) in the database [Rennhackkamp, 1997]. However, these implementations are insufficient. The data are stored inside the BLOB

as a noninterpreted byte stream. Because of this overly simple generalization, the DBMS does not have any knowledge concerning the content of the BLOB or its internal structure. This, in turn, precludes the storage of even moderately large imagery and results in poor system performance and memory consumption. Therefore, it is very important to store large images and their metadata with other GIS data in an integrated database [Zhou, 1995].

Indexing is another issue in large imagery management. The properties of large image content - such as textures, shapes, edges, and locations of objects are usually represented by multidimensional points, or spatial data. Traditional manually based text indexing or B- trees, that uses one-dimensional ordering of key values do not work in image database systems as they cannot describe the complex property of large imagery. Index structures that are appropriate for content based query and similarity based query have to be developed to fit the needs of image database systems. Further discussion about large imagery management issue will be in Chapter 2.

1.4.2 Large Imagery Query and Retrieval Issues

As stated above, the properties of large image content - such as textures, shapes, edges, and locations of objects are usually represented by multidimensional points, or spatial data. This means that large image database systems handle query retrieval methods based on similarity and content, as opposed to an exact match which is the text-based method used by almost all commercial relational database systems. Therefore, how to design query and retrieval methods that support similarity based, content based and non-exact queries, in addition to traditional text based queries and other GIS queries and retrievals, is an important issue to be

addressed by large imagery database systems.

In order to support effective and efficient query by similarity and by content in large imagery databases, in addition to traditional relational operations and GIS spatial and temporal operations, such as near, east, west, left and right, etc., more complicated information processing methods such as large imagery enhancement, classification and pattern extraction must be added to get the similarity and content information from the large imagery. The data processing issue becomes more serious when queries and retrievals are performed on a database with a large spatial extent, having multiple sources, and involving multispectral and multiresolution data. Some techniques, such as data abstraction and rules and user defined functions have to be designed and implemented to make large imagery databases more dynamic to meet the various application specific requirements.

1.5 Research Objectives

The main objective of this research is to study how to effectively and efficiently store, manipulate, query and retrieve large imagery, which is one of the important components of GISs, in a relational database. The basic idea is firstly, using advanced database management techniques, which include metadata management, indexing and clustering techniques, to make a large imagery database model. Secondly, advanced large imagery query methods such as query by similarity and query by content, are developed with powerful image preprocessing functionality. Finally, using this database model, a large imagery database prototype is implemented using the Oracle8™ database management system. This prototype has the following abilities:

1) to effectively and efficiently manage large imagery and its metadata in a single database.
2) to improve performance by using advanced compressing, indexing, partitioning and clustering techniques,
3) to allow users to manage a database without knowing any SQL syntax,
4) to allow users to add easily new functions and their knowledge into a function library to improve database performance,
5) to perform database browsing, query navigation, similarity and content based queries and retrievals.

1.6 Thesis Outline

In addition to this introduction, this thesis contains five chapters.

Chapter two discusses the existing problems of large imagery management in relational databases. It also covers proposed techniques for large imagery storage, compression, metadata management, data compressing, indexing, partitioning and clustering.

Chapter three discusses the large imagery preprocessing techniques that are very critical and important procedures for large imagery queries since they will significantly affect the query results. This chapter focuses on image classification, filtering processes that extract the polygons of objects from raw images, an image similarity measure calculation method and image content calculation method.

Chapter four firstly overviews traditional imagery query methods that are usually used for query and retrieval of medical images, art images, photos, etc., and then discusses database

browsing and suggests two new methods -- Query by Similarity (QBS) and Query by Content (QBC). These are both important and mandatory for satellite imagery and airborne imagery query and retrieval. The limitations of QBS and QBC methods are also discussed in this chapter.

Chapter five implements the techniques discussed above in a large image database prototype using Oracle8™ as its base. Test results and analyses on a group of large test images are provided. The discussion of the advantages and disadvantages of this prototype concludes this chapter.

Chapter six includes the main conclusions obtained from this project and also gives some recommendations for future developments related to large imagery support.

CHAPTER 2 LARGE IMAGE MANAGEMENT IN RDBs

2.1 Introduction

Relational database technology has proved to be successful at supporting standard data processing applications, such as accounting, inventory and payroll etc. This success is due to the mathematical simplicity of the relational model. However, relational databases are inefficient for supporting complicated objects, such as large imagery because of its large data volume, intensive processing requirements and rich semantics. From a data management point of view, there are many problems that need to be solved in large imagery management, such as the lack of user-defined data type definition capability, inefficient indexing methods, storage problems, data format conversion problems and data compressing problems etc. Among these problems, three of them are relevant to this research in large imagery management: the storage, query and retrieval and manipulation of large imagery. The solutions to these three problems are essential and critical to large imagery management. This chapter will focus on the investigation and development of large imagery storage, indexing and manipulation techniques. Chapters 3 and 4 will discuss query and retrieval techniques for large imagery.

Traditionally, either a relational database is used to store attribute data and metadata for large imagery while storing the raw images in flat files, or BLOBs (including long field, bulk data) is used to store large imagery as long un-interpreted byte strings in the same database as

alphanumeric data. Examples of these approaches can be found in Zhou et al. [1991], Biliris [1992], Stonebraker and Olson [1993]. Researchers and developers in image processing, computer vision and remote sensing communities have concentrated on working with a few large images via flat files with little concern for the management of substantial volumes of large images. On the other hand, the GIS and database communities have tended to treat large imagery as a "black box" using long fields, bulk data or BLOBs without much concern about their semantics, contents, data abstraction and processing techniques.

In the last several years, an evolutionary technology, known as Object/relational database management systems have appeared. These make it possible to store large imagery in relational databases. ORDBMSs add new object storage capabilities to the relational systems at the core of modern information systems. These new facilities integrate the management of traditional field data, complex objects such as time-series and geospatial data, and diverse binary media such as audio, video, images, and applets. By encapsulating methods with data structures, an ORDBMS server can execute complex analytical and data manipulation operations to search and transform multimedia, image and other complex objects.

The most important new object/relational features are user-defined types (UDTs), user-defined functions (UDFs), and optimizer enhancements -- that support UDTs and UDFs [Rennhackkamp, 1997]. UDTs include complex data types that may encapsulate complex internal structures and attributes. UDFs define the methods by which applications can create, manipulate, and access the data stored in these new data types. Users and applications simply call the UDFs and don't need to understand their internal structure. UDFs also support the

notion of overloading, which refers to the concept of using the same name for different routines (actually called methods or member functions). User-defined access methods define the techniques by which the UDTs are accessed and indexed. An extensible optimizer provides ways to assign costs to UDFs and user-defined access methods, so the DBMS can determine the best way to access the data stored in the UDTs. Examples of commercial ODBMS can be found in Rennhackkamp [1997].

Although ODBMS do provide the architecture to support complex objects, there are still problems in large imagery metadata management, indexing methods and query and retrieval methods. These problems will be discussed in the next several sections. These limitations not only degrade the system performance, but also make application development unnecessarily complicated.

In this chapter, firstly the existing problems associated with large imagery management in commercial RDBMSs will be discussed. Secondly, an efficient large imagery metadata management method is proposed. Finally, advanced indexing, image compression, partitioning and clustering techniques that are essential to large imagery management are investigated and developed.

2.2 Existing Problems

Zhou [1995] has given us a very complete overview on existing problems of large object management in GISs. Some of these are also associated with large imagery management in commercial relational databases. The major problems are listed in the following:

1) Often large imagery is not integrated with alphanumeric data. Images are often stored in different locations using different databases, flat files and structured data storage. This separation often makes data management problematic.

2) Some relational DBMSs store large imagery inside the BLOB as a noninterpreted byte stream. Because of this overly simple generalization, the DBMS does not have any knowledge concerning the content of the BLOB or its internal structure. Consequently, you cannot perform queries and operations on inherently rich and structured data types, such as images, video, Web pages, hypertext, and word-processing documents. The operations and algorithms to manipulate these data types are not available to the query processing and indexing facilities of the DBMS. This means that the users' application programs must perform the necessary processing on the contents of the BLOBs. This, in turn, suggests that the entire BLOB must be shipped across the network to the client's workstation before any operations can be performed on it. The BLOB may have to be shipped all this way, consuming a lot of unnecessary bandwidth, only for the application to determine that it has no interest in its content.

3) Data abstraction techniques such as multiresolution, data approximation and data partitioning are not used in large imagery management.

4) The management of metadata and other derived data of large imagery are neither effectively managed, nor meet the metadata standards. This results in data conversion and distribution being ineffective and error-prone.

5) Automatic or semi-automatic feature extraction techniques, which are essential to content based queries of large imagery, are often not available or have not been integrated into

DBMSs.

6) Traditional indexing methods are not appropriate for content based queries of large imagery because queries are issued to retrieve images by content. For example, given an object such as Lake Louise, one might ask to find all images that contain "Lake Louise". To perform this kind of queries requires an index on the results of a classification or feature extraction rather than on the raw images.

2.3 Metadata Management

2.3.1 Defining Metadata

Metadata or "data about data" describe the content, quality, condition, and other characteristics of data. Metadata are used to provide documentation for data products. In essence, metadata answer **who**, **what**, **when**, **where**, **why**, and **how** about every facet of the data that are being documented. This means that the relationship between a data object and the metadata describing it is functionally identical to the relationship between a book and its library catalog record. Another, for example, might be that of a student and his/her personal information stored in a department's or university's information management system, such as the student's ID, registration date, contract information and exam scores, etc.

Professionals from different areas have different types and different scopes of metadata. For example, a social science data archivist might use the term to refer to the systems and research documentation necessary to run and interpret a magnetic tape containing raw data. An electronic records archivist might use the term to refer to all the contextual, processing,

and use information needed to identify and document the scope, authenticity and integrity of a record in an electronic system. These diverse perspectives result in a very broad conception of metadata and reflect the complexity and difficulties of metadata creation and management. Therefore, it is very important to develop field-oriented metadata standards.

2.3.2 Why are Metadata Important?

Metadata play a very important role in large imagery management because of the following reasons:

1) They increase data accessibility. Effectiveness of searching can be significantly enhanced through the existence of rich, consistent metadata. Metadata can also make it possible to search across multiple data collections or to create virtual collections from data that are distributed across several sites and systems, but only if the descriptive metadata are the same or can be mapped to a consistent basis. This is very important and basic to data distribution and data warehouse development.
2) They improve the productivity of administrators and the reliability of solutions. Many components are used when building a system supporting Business Intelligence. They include database management systems, modeling tools, transformation tools, process managers, data mining and decision support tools, etc. These tools leverage many different platforms, data formats, and vendor suppliers. Furthermore, each tool typically has its own metadata store and administrative interface. Making a mix of tools work together requires passing data between them. Administrators must create and maintain such data bridges. Often, identical information must be fed to multiple tools creating for

data entry problems.

3) They assist end users in locating and understanding data. For example, in a large satellite image database, there might be hundreds of thousands of images and related information stored and managed by DBMS. Good metadata, such as the source of image, location of image storage can help users to locate the source of data; date and time of image capture, the solar illumination and cloud situation, etc. can help users understand this image better; histograms and other statistic values can help users extract the contents of an image.

As GISs are applied to an increasing assortment of environmental and development issues and spatial data sharing becomes more prevalent, the importance of metadata is spreading beyond the domain of the federal government to other GIS user communities at the local government levels, as well as non-profit organizations and the private sector. In 1994, the National States Geographic Information Council (NSGIC) of the U.S. was given a Federal Geographic Data Committee (FGDC) Competitive Cooperative Agreements Program (CCAP) award to undertake an educational and research program in support of content standards for digital geospatial metadata. The purpose of the standard is to provide a common set of terminology and definitions for documentation related to these metadata. This standard has been widely applied by the GIS community for geospatial data management, distribution and sharing. However, remote sensing, digital image processing, computer vision and other fields continue to work on their specific metadata standards.

2.4 Remotely Sensed Image Metadata

Remotely sensed images include satellite and airborne images. They are collected by sensors, loaded on satellites and airplanes, which have multiple channels in different wavelengths. They have to be preprocessed before they can be used by various applications. After the data are transmitted to ground stations from satellites or airplanes, the measurements are calibrated and corrections are applied. Next the data are corrected geometrically to reduce distortion, and location information is added. The data are then released for further processing [Lillesand et al., 1987]. Digital remotely sensed image data are an important data source for many applications, such as environmental assessment and monitoring, global change detection and monitoring, agriculture, nonrenewable resource exploration and mapping.

These remotely sensed images are perceived differently by computer and Earth scientists. The computer scientist focuses on structure and treats each image as a three-dimensional raster (a regular grid). For example, the dimensions for a TM data set are almost 6000 rows x 7000 columns x 7 deep. The Earth scientist, however, focuses on the processes that created the image. From his or her point of view, the image has 7 bands or layers, each created by a different process. Each band is composed of 6000 lines and 7000 samples. The database metadata schema should support both viewpoints.

The proposed remotely sensed image metadata scheme uses the Spatial Archive and Interchange Format (SAIF) [Surveys and Resource Mapping Branch, 1987] and the Content Standards for Digital Geospatial Metadata, which were adopted by FGDC to label geospatial

datasets [Federal Geographic Data Committee, 1997], standards as starting points. SAIF and FGDC are among the contributors to the Open Geodata Interoperability Specification (OGIS). SAIF is an extensible, object-oriented data modeling and archival/transfer standard that targets both geospatial and non-geospatial data. The Federal Geographic Data Committee (FGDC) effort to specify a digital geospatial metadata standard began in 1992, resulting in a final draft published in 1994. It adopts a simpler data model organized into a hierarchy of data and compound elements needed to document a geospatial data set. The standard provide a common set of terminology and definitions for the documentation of geospatial data. SAIF is highly extensible and committed to integration with other standards. The FGDC metadata standard is particularly useful for high-level metadata and appears to be gaining popularity in the geospatial user community.

A large image database based upon SAIF and FGDC metadata standards will be compatible with other databases or data warehouses which use the same metadata standard. This is very important and is mandatory for future work in areas like data distribution, data sharing and web-based large image database implementation.

Table 2-1 summarizes a SAIF view of our sample TM metadata. It is divided into the following parts: Grid, StructuredData, GeneralLocation, TimeObject, and SpatialReferencing. SAIF types are listed in the left column; compound types are italicized; attributes are in regular type; and the sample TM metadata are listed in the right column.

Grid	
Position_of_origin	-2050000, 752000
Grid_reference:	
Ordering	rowOrder
Origin_corner	nw
Cell_position_at_corner	false
Origin_coordinates	0, 0
Grid_framework:	
Grid_dimensions	6000, 7000
Grid_spacing	1000, 1000
Units	meters, meters
Offset_structure	none
StructuredData	
Content_values	actual image
Content_structure:	
Indexing_scheme	SequentialArray
Attribute_names	TM channels 1-7, satellite zenith, solar zenith, relative azimuth, pixel source date
GeneralLocation	
Bounding_box:	
Min_point	-119.9722899,23.5837576
Max_point	-65.3946489, 46.7048989
TimeObject	
Duration:	
Start_time	4-jan-1985
Length_of_time	14
SpatialReferencing	
Earth_model	Clarke 1866
Projection	UTM
Central_meridian	-100, 45
False_easting	0
False_northing	0

Table 2-1 SAIF Raster Image Metadata

The SAIF standard defines the basic metadata that are both important and sufficient to answer text-based queries about large imagery. In order to support advanced large imagery

queries, such as Query by Similarity and Query by Content, other image attributes will also be needed, such as statistical information and object information in addition to the basic information listed above. Table 2-2 shows an extended schema of the SAIF metadata standard that is used in the proposed large imagery database.

Grid	
Position_of_origin	-2050000, 752000
Grid_Reference:	
ordering	rowOrder
Origin_corner	nw
Cell_position_at_corner	false
Origin_coordinates	0, 0
Grid_framework:	
Grid_dimensions	6000, 7000
Grid_spacing	1000, 1000
units	meters, meters
Offset_structure	none
StructuredData	
Content_values	actual image
Content_structure:	
Indexing_scheme	SequentialArray
Attribute_names	TM channels 1-7, satellite zenith, solar zenith, relative azimuth, pixel source date
Bit	8
GeneralLocation	
Bounding_box:	
Min_point	-119.9722899,23.5837576
Max_point	-65.3946489, 46.7048989
TimeObject	
Duration:	
Start_time	4-jan-1985
Length_of_time	14
SpatialReferencing	
Earth_model	Clarke 1866
Projection	UTM
Central_meridian	-100, 45
False_easting	0
False_northing	0
StatisticalData	

Mean	57.6
Variance	19.2
Histogram	556,75,67568,...
Lorenz_measure	0.56,0.06,0.78,...
Data_quality	8
DataProcessing	
Preprocessing:	
Methods	Atmospheric correction,etc.
Image_processing:	
Methods	Contrast Enhancement, Object Matching and etc.
ImageObjects	
Object_name	Lake Claire
Area	229.576 square km
Perimeter	117 km
MBR_area	357.66 square km
Centeroid	-117.3511756,20.4928476

Table 2-2 Proposed Remotely Sensed Image Metadata Schema

These metadata are dependent on large images, and should be treated as such. Therefore, in order to maintain data integrity, when a large image is deleted from the database, the related metadata should also be removed from the database unless users request not to do so through a user defined method.

Due to the complexity and rich semantics of large imagery, it is impossible to pre-define and pre-calculate all metadata before permitting database queries. Therefore, it is necessary and important for a proposed database to support virtual metadata which will be defined, calculated and stored dynamically at the time of database queries and applications. Virtual metadata can be created by invoking related functions during a query session. For example, if the histogram of an image isn't stored in database, but is needed for Query by Similarity, a

calculate_histogram( ) function can be invoked and the histogram of this image can be calculated and stored in the database. This allows the system to be extended dramatically and many application-specific requirements can be supported easily.

2.5 Indexing

Unlike traditional database systems that manage text data only, large image database systems handle multidimensional point data and similarity or content based retrievals. Feature extractors compute a set of features which describe the properties of image content - such as colors, textures, shapes, edges, and locations of objects. Features are usually represented by multidimensional points, or by spatial data. Structures used in traditional database systems, such as B-trees, use one-dimensional ordering of key values that do not work in large image database systems as they cannot describe the complex property of large imagery.

Moreover, large image database systems handle retrieval methods based on similarity or content as opposed to exact match. For example, images can be requested whose colors are similar to the color of an indicated image or those images that contain the given object, for example Lake Louise. The most popular, probably the easiest, similarity function is based on Euclidean distance. In order to compare the similarity of images, image database systems compute the Euclidean distances of spatial data that represent the images' content.

As mentioned before, one-dimensional index structures do not fit the need of large image database systems. Several index structures have been proposed handle spatial data, i.e. multidimensional point data. This kind of index structure is usually called content-based

index structure because the queries are based on properties of image content. These indexing methods can be grouped into the following classes [Petrakis, et al., 1994]:

1) Methods that transform rectangles into points in a higher dimensional space.
2) Methods that use linear quad-trees or, equivalently, the "z-ordering" or other "space filling curves".
3) Methods based on trees (k-d-trees, hB-trees and cell-trees, etc.). One of the most characteristic and popular approaches in this class is the R-tree.

In this work, the R-tree is used as the underlying method for indexing large images by content. The reason for this choice is that the R-tree is more robust in high-dimensionality address spaces [Petrakis, et al., 1994].

A R-Tree, proposed by Antonin Guttman [Guttman, 1984], is an index structure for point and spatial data at the same time. Insert, delete and search can be intermixed without periodic reorganization. It uses a tuple to represent a spatial data in the database. In order to retrieve the data, each tuple has a unique identifier, tuple-identifier. At the leaf node of a R-Tree, it has an index record that can reference the spatial data. The index record is (I, tuple-identifier). I is a n-dimensional rectangle and it is the bounding rectangle of the spatial data indexed. This rectangle is also known as minimal bounding rectangle (MBR) (Figure 2-1) and each entry in tuple-identifier is the upper and lower bounds, [upper, lower], of the rectangle along the dimension. Non-leaf nodes contain entries (I, childnode-pointer) where I is the minimal rectangle bounding all the rectangles in the lower nodes' entries. Childnode-pointer is the pointer to a lower node in the R-Tree. Let M and m<=M/2 be the maximum and

minimum number of entries can be filled into one node respectively. [Fu, undated]

A R-Tree satisfies the following properties:

1) A R-Tree is a height balance tree and all leaves are on the same level.
2) Root node has at least two children unless it is the leaf node.
3) Every non-leaf node contains between m and M entries unless it is the root. For each entry (I, childnode-pointer) in a non-leaf node, I is the smallest rectangle that spatially contains all rectangles in its child nodes.
4) Every leaf node contains between m and M index records unless it is the root. For each index record (I, tuple-identifier) in a leaf node, I is the smallest rectangle that spatially contains the n-dimensional data object represented by the indicated tuple.

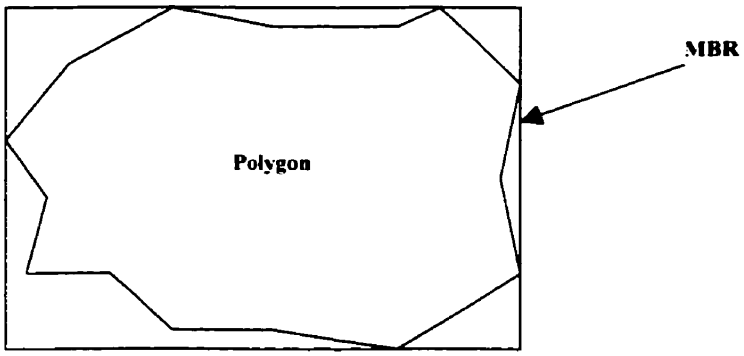


Figure 2-1 MBR of a Polygon

Figure 2-2 illustrates data rectangles (in gray) organized in an R-tree (left) and the file structure for the same R-tree is also shown (right); the nodes correspond to disk pages.

In Chapter 3, a set of object indexing algorithms will be developed that can be used to generate indices of objects for similarity and content based query and retrieval. Such algorithms include image classification, feature extraction and pattern recognition. Other methods such as those which combine color, shape and texture in large images to extract a group of parameters of an object can also be useful for extracting image contents [Petrakis, et al., 1994].

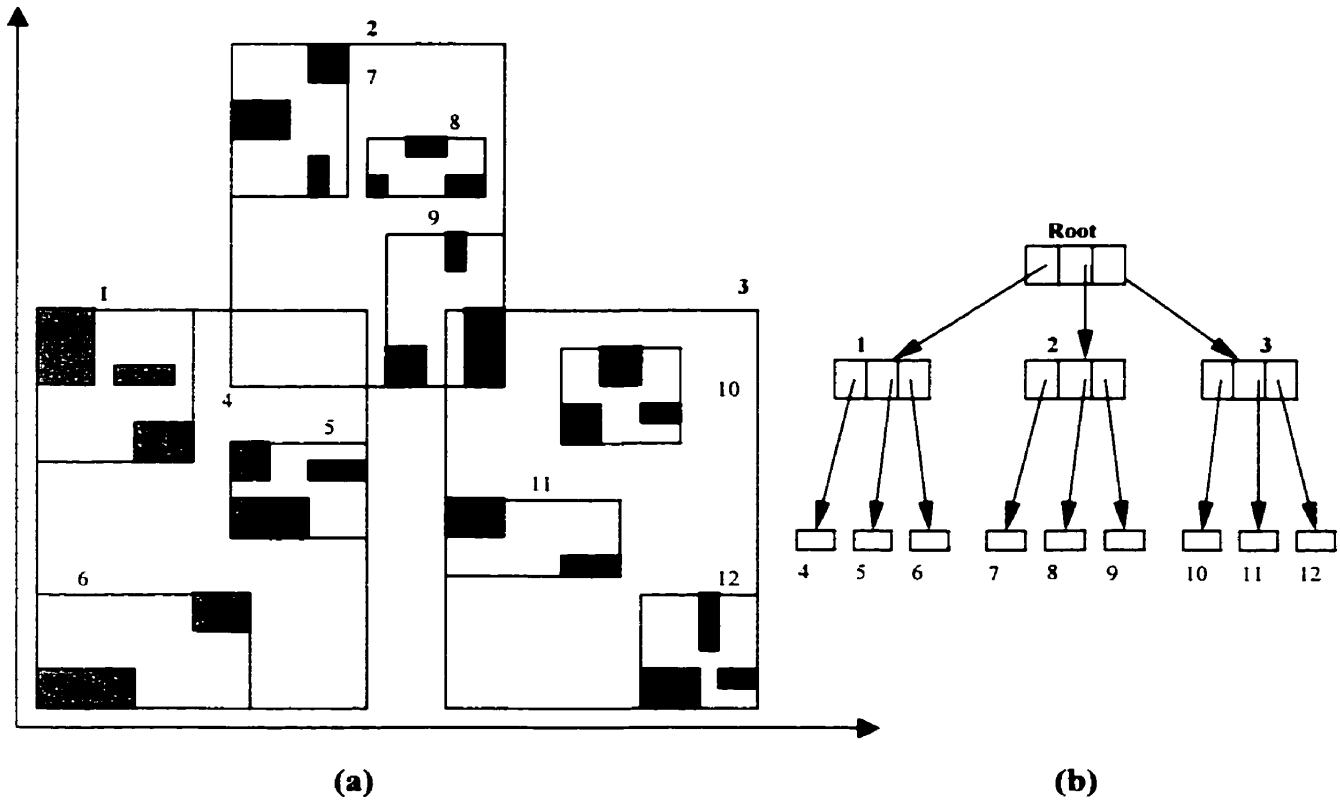


Figure 2-2 R-tree Illustrations

(a) Data (gray rectangles) Organized in an R-tree, (b) The Resulting Tree on Disk (adapted from [Petrakis, et al., 1994].)

2.6 Image Compression

One of the important characteristics of large image is the huge volume of data. In practice, large amounts of data result in difficult storage, processing, and communications requirements. Image compression addresses this problem by taking advantage of patterns in a large image. The underlying basis is the removal of redundant data and in effect, squeezing the data to maximize the information contained in each byte. Satellite images, scanned maps and digital photos are usually considered as good candidates for compression. By using appropriate image compression methods for large images, these sizes can be significantly reduced and a large image database can transfer more image information with fewer bytes, thus making more effective use of available bandwidth. Undoubtedly, compression is one of the most critical issues that should be considered when designing a large image database because it can significantly affect the data storage, image management, query and retrieval speed, data distribution and communication, etc.

Interest in image compression dates back more than 25 years and many well-known compression algorithms have been developed over the years, such as run-length coding, Huffman coding, TIFF, GIF, JPEG, MPEG and wavelet, etc. Detailed study of image compression algorithms is beyond the scope of this research but has been discussed by many researchers [Gonzalez, 1992; Levine, 1994; Jahne, 1991]. It is important to note that different images will have different requirements for data compression, depending on the intended applications. In this research, Oracle8™ is employed as the RDBMS and Oracle Developer 2000 is employed as development tool which supports almost all of the popular graphics file

compression algorithms. Table 2-3 shows the image file format currently supported by Oracle8™.

It can be concluded from this table that Oracle8™ supports lots of image compression methods, which is critical to large image databases because they are supposed to handle various image formats.

Format	*Feature*	*Compression*
BMP	Monochrome, 4 and 8-bit LUT, 24-bit RGB	none
JFIF	24-bit RGB	JPEG
PCX	Monochrome; 2, 4, and 8-bit LUT; 1, 2, and 8-bit RGB	RLE
PICT 1&2	Monochrome; 2, 4, and 8-bit LUT; 16 and 24-bit RGB, vector/object graphics	Packbits
GIF	8 bit LUT	LZW
CALS	Monochrome	CCITT G4 (FAX)
PCD	Monochrome, 4 and 8-bit LUT, 24-bit RGB	(Kodak)
RAS	Monochrome, 4 and 8-bit LUT, 24-bit RGB	RLW
TIFF 4.5, &6	Monochrome, 8-bit gray, 4 and 8-bit LUT, 24-bit RGB. Planar data, Tiled data, Intel byte order, Motorola byte order, Photometric interpretation, MSB/LSB	Packbits, CCITT G3(FAX), CCITT G4 (FAX), LZW, LZW with horizontal difference, JPEG

Table 2-3 The Image Compression Methods Supported by Oracle8™ (adapted from Oracle [1998])

2.7 Clustering and Partitioning

Large image databases must deal with not only images, but also handle other data such as image metadata, related text, numeric and graphical data, or in other words, heterogeneous data. Therefore, heterogeneous data are one of the main characteristics of large image databases. In order to make this database more effective and efficient, some special database

techniques such as clustering and partitioning have to be implemented.

2.7.1 Clustering

Clustering is a database technique used to store a group of objects physically close together so that they can be retrieved more efficiently [Zhou, 1995]. Clustering refers to the process of grouping together related objects whether at the conceptual level, in main memory or in secondary storage. The main purpose for clustering is to increase the probability that when an object is accessed by a user's procedure, objects that it references or is related to, can be accessed quickly by virtue of being on the same or neighbouring physical storage block.

In relational database management systems, clustering groups tables that share the same data blocks because they share common columns and are often used together. For example, the EMP and DEPT tables share the DEPTNO column (Figure 2-4) [Oracle, 1998]. When you cluster the EMP and DEPT, RDB physically stores all rows for each department from both the EMP and DEPT tables in the same data blocks.

Because clusters store related rows of different tables together in the same data blocks, clustering offer two primary benefits:

- Disk I/O is reduced and access time improves for joins of clustered tables.
- In a cluster, a cluster key value is the value of the cluster key columns for a particular row. Each cluster key value is stored only once in the cluster and the cluster index, no matter how many rows of different tables contain the value.

Therefore, less storage might be required to store related table and index data in a cluster than is necessary in nonclustered table format. For example, in Figure 2-3 notice how each cluster key (each DEPTNO) is stored just once for many rows that contain the same value in both the EMP and DEPT tables. Apparently, this will significantly improve storage and retrieval performance for large image databases. In this research, since image raw data, metadata and description data are closely related, object clustering based on image raw data is selected. Image raw data are stored as objects with their object ids. The identity of a large image is created, stored and managed by RDBMS to provide efficient access to large images. Because the metadata of a large image is often accessed at the same time as other attribute data, they are explicitly stored in the same data blocks to improve the data access performance.

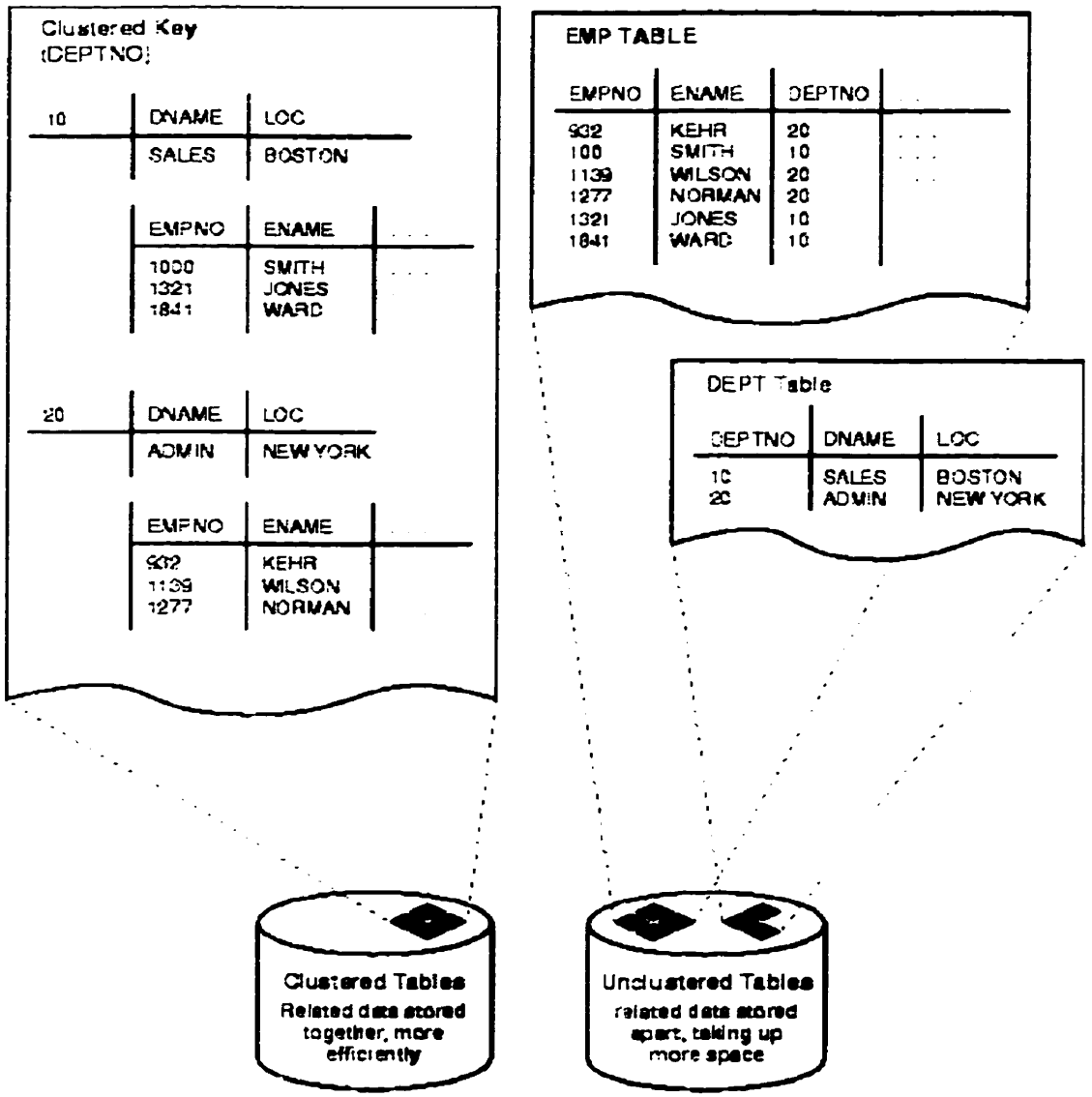


Figure 2-3 Comparison of Clustered Table and Nonclustered Table

(adapted from Oracle [1996])

2.7.2 Partitioning

In modern RDBMSs, partitioning addresses the key problem of supporting very large tables and indexes by allowing the users to decompose them into smaller and more manageable pieces called partitions. For example, image raw data, related metadata and attribute data for each state or province could be stored in a separate partition of a large table. Data related to a particular state would be physically clustered for faster storage and retrieval. Once partitions are defined, SQL statements can access and manipulate the partitions rather than entire tables or indexes. Partitions are especially useful in a data warehouse or large image database applications which commonly store and analyze large amounts of historical or image data.

A large image database has to deal with large amounts of image data with rich semantics. It could benefit greatly, as described below, from the use of partitioning [Oracle, 1998]:

1) Partitioning can increase the availability of large data set if critical tables and indexes are divided into partitions to reduce the maintenance windows, recovery times, and impact of failures. The users can also improve access performance to a critical table or index by controlling performance parameters on a partition basis.
2) Partitions enable data management operations for very large database operations like data loads, index creation, and data purges at the partition level, rather than on the entire table, resulting in significantly reduced times for these operations. Partitioning can significantly reduce the impact of scheduled downtime for maintenance operations by introducing partition maintenance operations that operate on an individual partition rather than on an entire table or index and by providing process independence so that maintenance

operations can be performed concurrently on different partitions.

3) Partitions help solve large image database performance problems. Since a large image database deals with large amounts of data, an ad-hoc query that requires a table scan can take a long time, because it must inspect every row in the table; there is no way to identify and skip subsets of irrelevant rows. Using partitioning, an ad-hoc query which only requires rows that correspond to a single partition (or range of partitions) can be executed using a partition scan rather than a table scan.

4) Partitioning can control how data is spread across physical devices. To balance I/O utilization, you can specify where to store the partitions of a table or index. With this level of location control, you can accommodate the special needs of applications that require fast response time by reducing disk contention and using faster devices. On the other hand, data that are accessed infrequently, such as old historical data, can be moved to slower devices or stored in subsystems that support a storage hierarchy.

There are several popular partitioning methods: range partitioning, which partitions the data in a table or index according to a range of values, and hash partitioning, which partitions the data according to a hash function. Another method, composite partitioning, partitions the data by range and further subdivides the data into subpartitions using a hash function. [Oracle Concepts, 1998] has given us a complete review of these methods. In this project, a table-based partitioning method, which is appropriate for spatial data, is employed. When used, this technique relieves database designers from having to anticipate how much data they will need to handle, how many tables they will need, or how to distribute the data effectively among many tables. If a table containing point data representing a specific region becomes

too dense and reaches a predetermined size, the table automatically subdivides into multiple partitions based on the data's spatial organization. The database designer needs only to supply a best estimate for a comfortable maximum partition size for a particular application. With the partitioning technique, the bigger the database, the more it may benefit.

2.8 Summary

This chapter discussed metadata management, storage management, indexing methods, clustering and partitioning techniques that are all critical and important issues for large imagery databases. Based on the nature and characteristics of remotely sensed images, a remotely sensed image metadata standard that standard was investigated and an extension of SAIF metadata standard for the proposed large image database was developed. R-tree, which is one of the most effective indexing methods to manage spatial data, was selected to be the indexing method for the proposed system. Modern database management techniques, image compression, clustering and partitioning were also investigated and employed to handle large images more effectively and efficiently.

CHAPTER 3 LARGE IMAGE PREPROCESSING IN RDBs

3.1 Introduction

Raw data of large imagery, for example, remotely sensed raw data, don't include any information of the image contents and metadata. To a computer and most people, one digital image is just a group of pixels each of which has a gray scale value, or we can say, an image is just a two-dimensional raster (a regular grid). When the users get images, especially remotely sensed images from ground stations, they usually only include information, such as the time and weather conditions when the image is taken, calibration and correction parameters, etc. There is little information about the features and objects the images contain. In order to perform QBS and QBC, large image preprocessing work has to be done in advance. The raw images have to be interpreted in detail and the objects they contain have to be identified. That is, more metadata, other than those mentioned above, such as image statistic analysis information, similarity measures and object attribute data, etc. have to be derived, indexed and stored in the image database.

Traditionally, image interpretation is done by involving skilled and experienced human analysts who locate, identify and label features of interest. With most large images now available in digital form, the use of computers for information extraction is standard practice. Many operations have been developed to extract the image features and content information, such as attribute data calculation, statistical analysis, spatial analysis, thematic classification, filtering, segmentation and feature extraction, etc. [Schowengerdt, 1997]. Though computer

tools using these operations speed and improve analysis, in most cases, visual interpretation cannot be supplanted completely by computer techniques. Also these operations can be difficult and very time consuming because many of them are complicated and require well designed and highly efficient implementation methods that incorporate application domain rules and knowledge.

Commercial relational database management systems do not provide many data and image analysis or processing functions, except for simple aggregation functions, because of the simplicity of the data types they support. Some image databases which are based on relational databases use other image processing programs or spatial analysis programs to do image processing work. In this way, these application programs will read image data from the database, process data and then write data and processing results back to the database. This approach requires various data conversions and spends much computer I/O time which complicates the database system and degrades system performance. In addition to these problems, it's also often very difficult or impossible for users to add new operations or change existing operations. Therefore, there is an obvious requirement to integrate image processing functions into relational database and let users easily extend these functions.

This chapter is devoted to the discussion of large image preprocessing techniques used for QBS and QBC. The following topics will be discussed: thematic classification and object extraction techniques, filtering techniques, similarity calculation methods and content extraction methods. The chapter is concluded with a summary.

3.2 Filtering Techniques

Filtering is one important part of image enhancement techniques. The principal objective of image enhancement techniques is to process an image so that the result is more suitable than the original image for a specific application. Here, the word specific is important, because the techniques are very much problem oriented. For example, a filtering method that is quite useful for enhancing x-ray images may not be the best approach for enhancing satellite images. Even a filtering method that is good for a satellite image which covers a city may not be suitable for a satellite image which covers a mountainous area. Therefore, in some cases, for example, the image that covers both city and mountainous area, before filtering is employed, the image should be partitioned which means the city area and the mountainous area should be segmented. Upon the partitioning is finished, the appropriate filtering methods should be used to city and mountainous area separately.

The filtering techniques fall into two categories: spatial domain methods and frequency domain methods [Gonzalez et al., 1992]. The spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. Frequency domain processing techniques are based on modifying the Fourier transform of an image.

Spatial filtering uses spatial masks for image processing, as opposed to frequency domain filtering using the Fourier transform. The masks themselves are called spatial filters. Spatial filters are classified into two categories: smoothing filters and sharpening filters. Smoothing filters are used for blurring and for noise reduction. Sharpening filters are used for

highlighting fine detail in an image or to enhance detail that has been blurred, either in error or as a natural effect of a particular method of image acquisition. Both smoothing and sharpening are important preprocessing techniques required for large image databases and are used in different image preprocessing procedures. For example, blurring or smoothing filtering is used in preprocessing steps, such as the removal of small details from an image prior to object extraction and bridging of small gaps in lines or curves while sharpening filtering is used to extract the edge of objects after image classification.

There are many books that discuss filtering techniques, such as Gonzalez et al.[1992] and Lillesand et al.[1987]. The detailed implementation of these filters is beyond the scope of this project. However, brief reviews of these filters help us to understand the techniques. Smoothing filters are also called lowpass filters including lowpass spatial filters and lowpass filters in frequency domain. Figure 3-1 shows an example of blurring by smoothing masks. Note in particular the loss of sharpness in the filament of the bulb. Popular sharpening filters include highpass spatial filters, high-boost filters and derivative filters, etc., Figure 3-2 shows an example of edge enhancement by derivative filtering. Note that the principal edges of the objects are enhanced considerably. This will be very useful for the next step classification and object extraction.

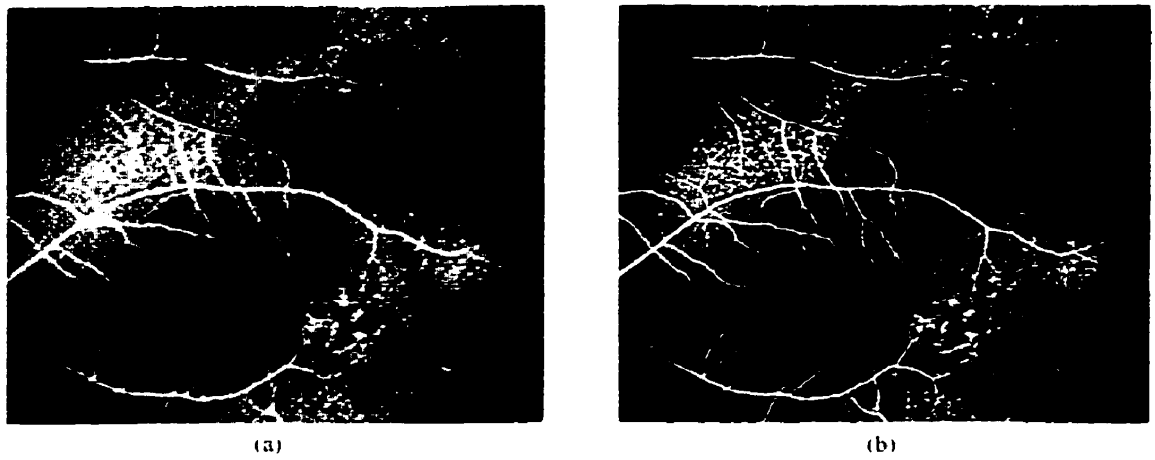
In terms of spatial filtering, enhancement in the frequency domain in principle is straightforward. We simply compute the Fourier transform of the image to be processed, multiply the result by a filter transfer function, and take the inverse transform to produce the enhanced image. Lowpass filtering in the frequency domain achieves image smoothing by

attenuating a specified range of high-frequency components in the transform of a given image. This is because edges and other sharp transitions, such as noise, in the gray levels of an image contribute significantly to the high-frequency content of its Fourier transform. Figure 3-3 shows the result of applying an ideal lowpass filter to the given image. Opposed to lowpass filtering, highpass filtering can achieve image sharpening in the frequency domain by attenuating the low-frequency components without disturbing high-frequency information in the Fourier transform. Figure 3-4 shows the result of using highpass filters in the frequency domain to a given image. Note how the object edges are extracted.

(a) **(b)**

Figure 3-1 Example of Lowpass Spatial Filtering:

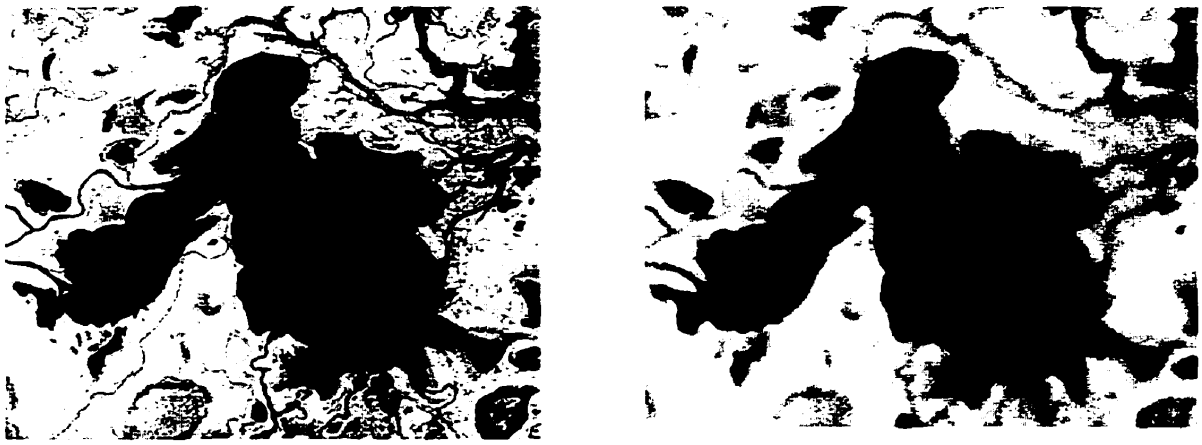
(a) Original Image (b) Resulting Image



(a) (b)

Figure 3-2 Example of Highpass Spatial Filtering:

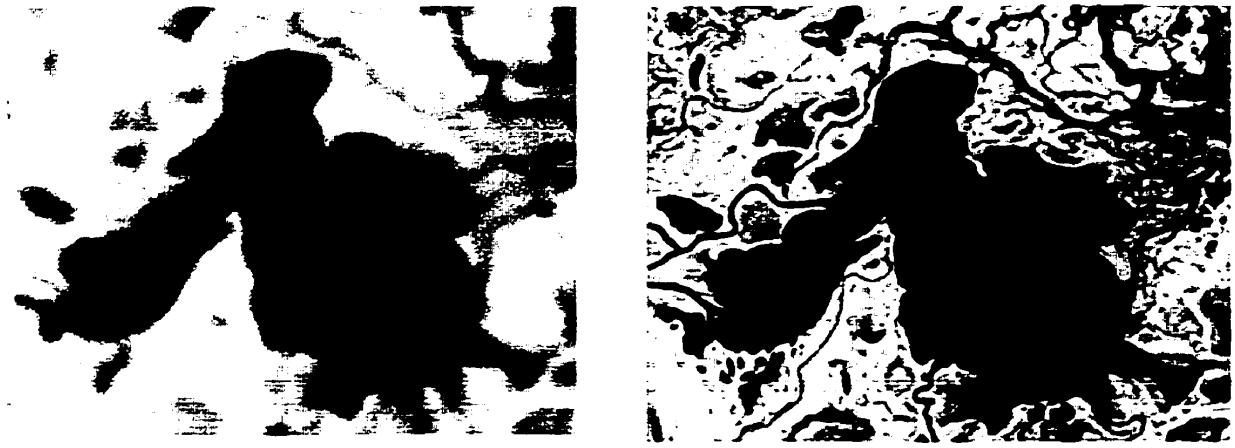
(a) Original Image, (b) Resulting Image (adapted from [Gonzalez, 1992])



(a) **(b)**

Figure 3-3 Example of Lowpass Filtering in Frequency Domain:

(a) Original Image (b) Resulting Image



(a) (b)

Figure 3-4 Example of Highpass Filtering in Frequency Domain:

(a) Original Image (b) Resulting Image

Many of these filtering methods are useful for thematic classification and object extraction which will be discussed in the next section. The fact is needed to be addressed that these approaches do not represent the addition of any new information to the image, but rather a redistribution of the original information into a more useful form.

3.3 Classification

Classification of large images is part of the most important procedure in object extraction because the results of these methods significantly affect the results of content based image queries and retrievals.

The overall objective of image classification is to automatically categorize all pixels in an

image into land cover classes or themes (i.e. provide an information description over an image, rather than a data description). Image classification, used mostly in remotely sensed image processing, is the process used to produce thematic maps, which shows the spatial distribution of identifiable Earth surface features, from raw raster imagery. The themes can range, for example, from categories such as soil, vegetation and surface water as a general description of a rural area, to different types of soil, vegetation and water depth or clarity for a more detailed description [Schowengerdt, 1997].

Traditionally, classification of an image involves several steps: feature extraction, training and labeling.

Feature Extraction

Feature extraction is the transformation of a multispectral image using a spatial or spectral transform to a feature image. Examples are selection of a subset of bands, a spatial smoothing filter. This step is optional, i.e. the multispectral image can be used directly, if desired. But the multispectral image contains all the various external influences, such as atmospheric scattering and topographic relief. Also, the data are often highly correlated between spectral bands, resulting in inefficient analysis. Furthermore, image-derived features, such as measures of spatial structure, may provide more useful information for classification. Thus, it is prudent to consider various preclassification transformations to extract the greatest amount of information from the original image. The filtering techniques described in last section are useful in that regard.

Training

In order to classify an image into categories of interest, the classification algorithm needs to be trained to distinguish those categories. Representative category samples, known as training samples, are used for this purpose. After the classifier is trained to recognize the different categories represented by the training samples, the "rules" that were developed during training are used to label all pixels in the image as one or more of the training categories.

The training of a classification algorithm can be either supervised, in which case the prototype pixel samples are already labeled by virtue of ground truth, existing maps, or photointerpretation, or unsupervised, in which the prototype pixels are not labeled, but have been determined to have distinguishing intrinsic data characteristics. In this research, unsupervised classification is selected to perform information retrieval from image raw data. The reason of this selection is that, in large image databases, there are thousands of images. Supervised training needs analysts to interactively select training samples and this task is time consuming and knowledge based. It's almost impossible and not practical for analysts to supervise each image classification interactively. Furthermore, an image database has to support dynamic image classification (i.e., in case that a new image without content information stored in the database is queried, it's required to retrieve new image classification results quickly). Therefore, there will just be a brief review of unsupervised classification and its algorithms in this section. For supervised classification techniques, it is suggested to refer to Schowengerdt [1997] and Lillesand [1987].

For unsupervised training, this research employs a computer algorithm that examines the unknown pixels in an image and aggregates them into a number of classes based on the natural groups or clusters present in the image values. The basic premise is that values within a given cover type should be close together in the measurement space, whereas data in different classes should be comparatively well separated.

There are numerous clustering algorithms that can be used to determine the natural spectral groupings present in a data set such as the K-means approach, image "texture" sensitivity algorithms, and maximum-likelihood algorithms, etc. These approaches have been discussed in a number of books such as [Schowengerdt,1997] and [Lillesand ,1987]. Among these algorithms, it is not possible to declare one algorithm better than another for all applications. Their performance is strongly data dependent. For example, the K-means approach is often applied only to image subareas rather than to full scenes because it is iterative, therefore, it is computationally intensive. If the class distributions of an image are Gaussian, the maximum-likelihood algorithm results in a minimum total misclassification error. The quality of the classification mainly depends upon the analyst's understanding of the concepts behind the methods and algorithms available and knowledge about the land cover types under analysis.

Upon finishing the classification, the resultant information, such as area, water clarity, other related information of soil, vegetation and surface water, is retrieved and stored in a database which can be used for image queries and retrieval such as "Looking for an image that contains an orchard whose area is bigger than $155m^2$ ". Also, classification results can be used for object extraction which will be discussed in Section 3.5.

3.4 Similarity Based Metadata Retrieval

Similarity retrieval in large image databases can be stated as: Given an image, find one or more images which are similar to the given image with respect to a given similarity measure. For example, given a TM image covering Banff National Park that was taken in June, find images in the database that are similar to this given image. From this definition, we can see that the given similarity measure is the most important factor in the similarity retrieval.

Many similarity measures that include statistics (covariance, correlation coefficient), image histogram, information theory (entropy), have been well developed and documented in the image processing literature [Gonzalez et al., 1992; Jahne, 1991; Pertrakis and Faloutsos, 1994]. In this research, a technique based on the Lorenz information measure of an image is chosen to implement a similarity measure on a large image. The techniques are based on the picture information measure proposed by Chang and Yang [1982].

Let $\{h(i)\colon i = 0,1,\ldots,L-1\}$ represent the histogram of a large image f, where $h(i)$ is the number of pixels with gray level i. The pictorial information measure PIM (f) is defined as follows:

$$PIM\ (f) = \sum_i h(i) - \max\ h(i) \qquad (3\text{-}1)$$

for $i = 0$ to L-1. Let the total number of pixels in f be N and $p_i = h(i)/N$, a generalized pictorial information measure NPIM (f) can be defined as follows:

$$NPIM\ (f) = 1 - \max\ p_i \qquad (3\text{-}2)$$

A more general pictorial information measure $NPIM_K(f)$ can also be defined as follows:

$$NPIM_K(f) = 1 - (\sum_i p_i) \qquad (3\text{-}3)$$

where i ranges over the k largest p_i's. Suppose the p_i's are ordered such that

$$p_0 <= p_1 <= .... <= p_{L-1}$$

and S_K is defined to be $NPIM_{L-K}(f)$, then

$$S_0 = 0,\ S_L = 1,\ S_K = \Sigma_i p_i,\ i = 0 \text{ to k-1}$$

which is called Lorenz information measure. By plotting the points (k, S_k), $k = 0, 1, ..., L-1$, a piecewise linear curve which is called a Lorenz curve can be obtained. This is illustrated in Figure 3-5. Noted that this curve represents the information content of a large image [Zhou, 1995].

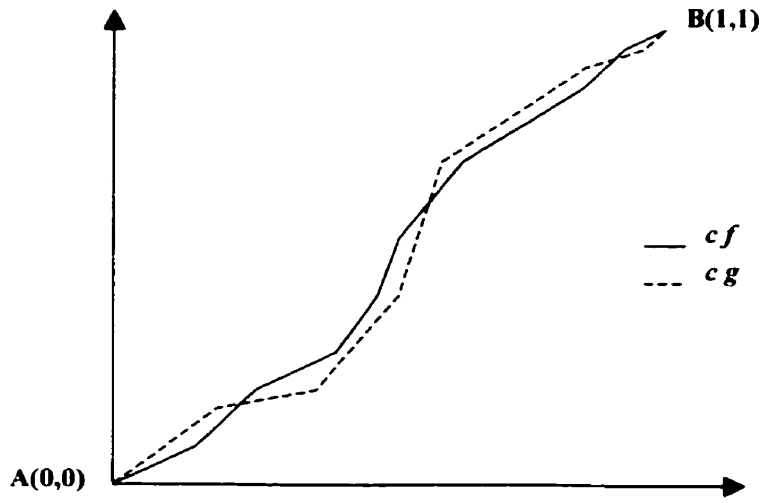


Figure 3-5 Lorenz Curves of Two Images

Because the Lorenz information curves are always normalized, curves for images having different sizes can be plotted and compared. In this project, a similarity measure between pictures *f* and *g*, *d (f, g)*, is defined as the summation of the polygonal areas enclosed by the two curves *cf* and *cg*. Obviously, 0 <= *d (f,g)* <= 0.5. The principle is if *d(f,g)* is below a preset threshold *t*, the two pictures can be considered as informationally similar in the sense of having similar Lorenz information curves.

There are several problems associated with this method:

- It is difficult to set the threshold value *t* and the method suffers various problems associated with exact-match retrieval techniques. A ranking function applied to the results in order to take into consideration of the fuzzy nature of the query and retrieval type is more appropriate. The original approach is improved by providing a weighting function to calculate the similarity values *s(f,g)*:

$$s(f,g) = w_h * \mathrm{var}(f,g) + w_d * d(f,g) \qquad (3\text{-}4)$$

  where *var(f,g)* is the normalized difference of variance between two histograms:

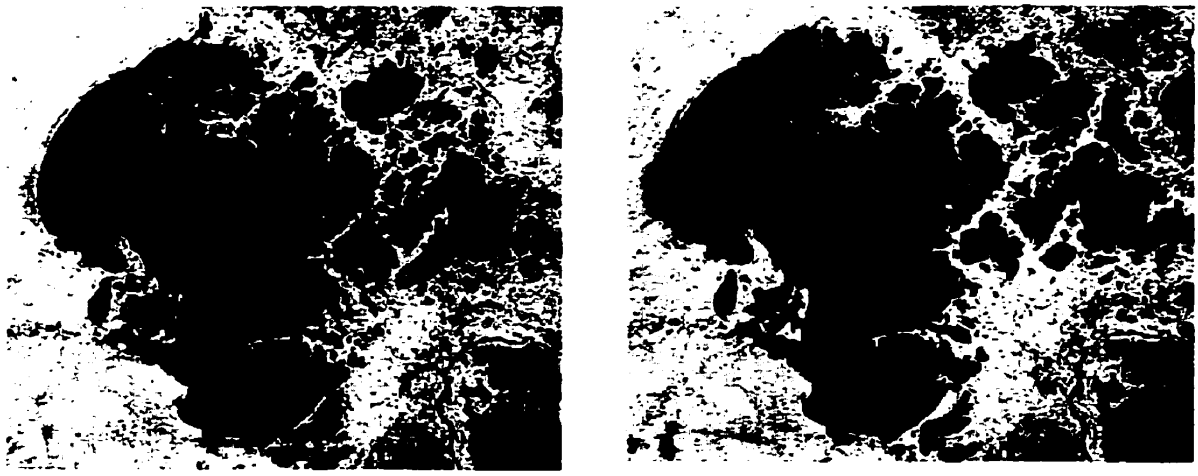
$$\mathrm{var}(f,g) = \frac{\max\{\mathrm{var}(f),\mathrm{var}(g)\} - \min\{\mathrm{var}(f),\mathrm{var}(g)\}}{\max\{\mathrm{var}(f),\mathrm{var}(g)\}} \qquad (3\text{-}5)$$

  w_h is the weight for *var(f,g)* and w_d is the weight for *d(f,g)*. They are set to 0.3 and 0.7 by default. A certain number of objects which have the lowest ranking orders are retrieved based on the value of similarity *s(f,g)*.

- It is likely that the background pixels are used in calculating similarity measures. The

background information may be totally irrelevant to the query and retrieval. This potential problem is solved by removing a preset percent of the lower portion of the histogram.

Figure 3-6 shows two images and the respective Lorenz measure values. From the values, it can be easily concluded that the two images are quite similar.



(a) Lorenz = 0.0277 **(b)** Lorenz = 0.0253

Figure 3-6 Similar Images Using the Lorenz Measure

In this research, when an image is stored in the database using the read_image ( ) function, the function calculate_Lorenz ( ) is invoked automatically. Lorenz values of this images are calculated and stored in the corresponding table ready to be used by similarity based query and retrieval.

3.5 Content Based Metadata Retrieval

Content based queries and retrievals mean finding the images that contain individual objects that are the same as the objects in the given image. For example, given Lake Louise, find the images in the database that contain Lake Louise. To support queries by image content, all images must be analyzed prior to storage so that descriptions of their contents can be extracted and stored in the database together with the original images. Therefore, this kind of query calls for techniques that can be applied to the contents of individual features inside a large image. The classification and similarity approaches discussed before will not work for each individual feature because they are based only on the statistical analysis results and not on the contents of images. Therefore, more objective measures, such as geometry, topology and the thematic information of features are needed.

The raw data of satellite and other images are just arrays of points with gray level values. They don't include any information about the individual object they contain. In order to derive the information about these objects, firstly object extraction techniques should be applied to the raw images to get the boundaries of these objects. This kind of technique is called object recognition or object extraction. In computer vision and image processing, lots of object extraction techniques have been discussed and implemented. In this research, a technique based on classification and edge detection has been discussed and implemented.

Figure 3-7 shows the processing step used in this research. It is assumed that the input data have been calibrated, geometrically corrected, radiometrically corrected and noise removal, strip removal and other preprocessing techniques have been applied. Then the first step in

classification, as described before, is to identify the pixels of interest. That is, the pixels represent the object to be extracted. Secondly, edge detection is used to extract the boundaries of the object of interest by detecting meaningful discontinuities in gray level. Thirdly, edge linking has to be applied to the results of edge detection. The reason of doing so is that, ideally, edge detection should yield pixels lying only on the boundary of objects. In practice, this set of pixels seldom characterizes a boundary completely because of noise, breaks in the boundary from nonuniform illumination, and other effects that introduce spurious intensity discontinuities. Thus, edge detection technique typically are followed by linking designed to assemble edge pixels into meaningful boundaries. The manual editing process is just included to correct potential problems existing in the automatic object extraction to guarantee that the closed boundaries of each object have been obtained.

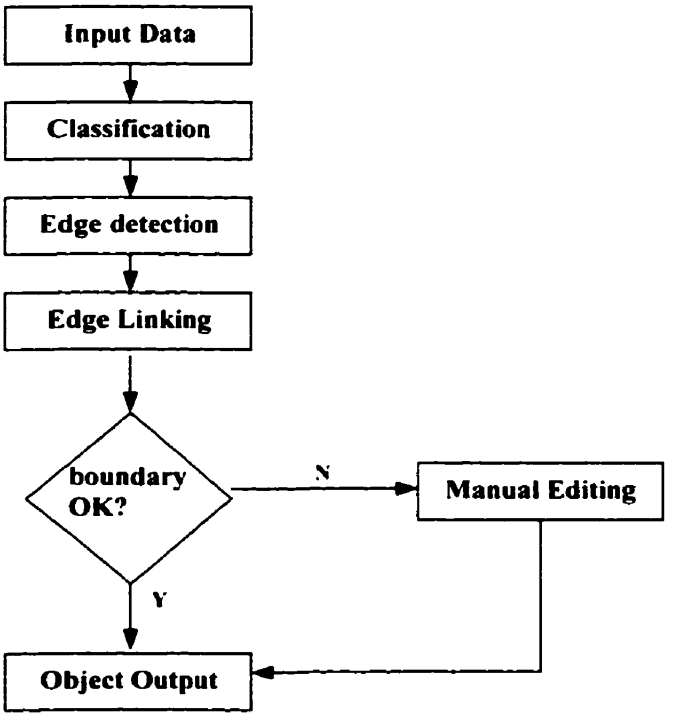


Figure 3-7 Processing Steps of the Object Extraction

The object extraction is one of the most important and sometimes most difficult tasks in large image query. This step determines the eventual success or failure of the Query by Content. In fact, effective object extraction rarely fails to lead to a successful query. In short, the better object extraction that can be done before the images are entered into the database, the better the query results can be expected. Upon finishing the object extraction, the content descriptions can be calculated and stored in the database together with the raw large images.

Content descriptions of large images are given in terms of object properties and in terms of relationships between objects. Such content descriptions are represented by relational structures of Content Relational Graphs (CRGs). Figure 3-8 shows an example image containing 3 objects and its corresponding CRG. Each object has a set of properties which describe the object and its relationship with other objects:

1) Area.
2) Perimeter.
3) Centroid (geometrical center).
4) Roundness: the ratio of the area of a polygon to the area of its MBR.
5) Distance: the distance of the centroid of the polygon to the centers of mass of other polygons in the same image.
6) Orientation: the angle between the horizontal direction and the longest segment of a polygon.

To make the above parameters both scale and translation invariant, the following processing methods are strongly recommended to be performed: Firstly, register all large images to a

common coordinate system. Secondly use the relative relationship instead of the absolute relationship of objects. These processing methods are related to multiresolution or multiscale issues. They will be discussed in detail in Chapter 4.

For each object in an image, the above properties are calculated and stored in the database. The center of MBR of each object is stored in R-tree nodes for object indexing and fast object retrieval. Finally, one CRG is generated for each large image and will be used for content based query and retrievals which will be discussed in the next chapter.

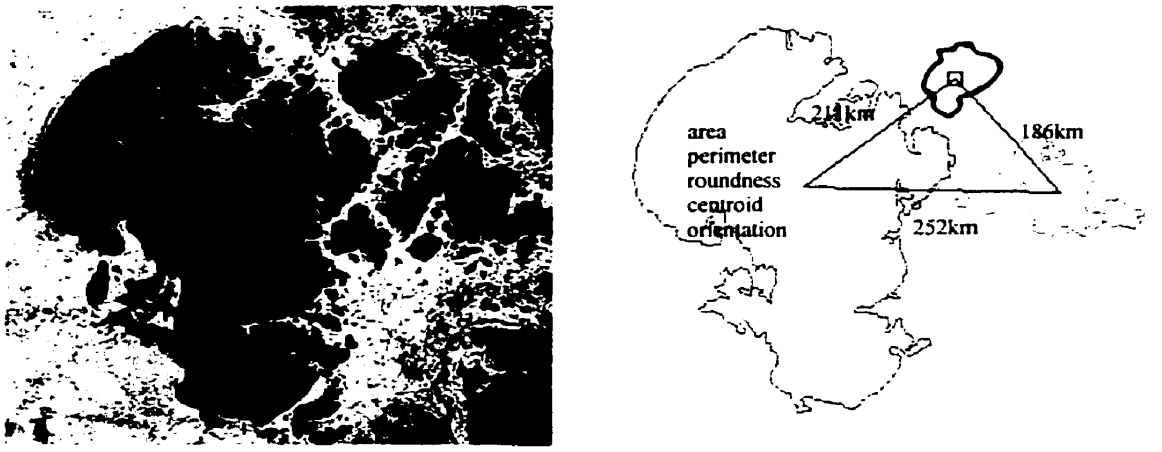


Figure 3-8 Example Image and Its Corresponding CRG

3.6 Summary

This chapter discussed the preprocessing techniques that are important to special query methods -- Query by Similarity and Query by Content of large images. To support similarity based queries and retrievals, similarity measures based on Lorenz information measure have been discussed and implemented. Also object extraction techniques based on classification

and edge detection have been designed and implemented. It has been shown that filtering techniques are critical to object extraction. Based on object extraction results, methods to obtain image content description parameters and a Content Relational Graph (CRG) for each image was also discussed.

CHAPTER 4 LARGE IMAGE QUERY IN RDBS

4.1 Introduction

Current technology allows us to generate, scan, transmit, store and manipulate large numbers of digital images. With the rapid increase in size of large image databases, the selection of an effective query and retrieval method is becoming more critical to image access. Traditionally, large image retrieval is based on captions, i.e. texts. Although useful in some cases, there are several problems with this approach, such as the fact that often the original keywords do not allow for unanticipated search in subsequent applications, and more important, inadequacy of uniform textual descriptions of such categories as color, shape and content, etc. Therefore, in large image applications, it widely calls for a query paradigm which can effectively support "like" style query and retrieval, such as QBS and QBC.

It is generally agreed that a query interface of a large image database should have the following features:

1) It should support similarity and content based queries and retrievals other than text based queries and retrievals.
2) The combination of visual query methods (e.g. QBS and QBC) and a text oriented query method is preferred to the one which is only text oriented, because the combination can provide both ease of use and the capability of the powerful query methods.
3) The user should be able to visually evaluate and refine the query results.
4) There should be a browsing facility.

5) The amount of information (e.g. the level of detail) at any particular point should be user controllable.

This chapter will mainly discuss how to make the query system of the proposed large image database meet the criteria described above. Firstly, Section 2 will overview the different image query types and difficulties of similarity and content based queries and retrievals. Database browsing techniques will also be discussed. Then the query processing methods of QBS and QBC will be discussed and implemented. Section 5 concludes this chapter with a summary of discussions and techniques presented in this chapter.

4.2 Types of Large Image Queries

As described before, in a traditional image database, large images can be accessed, queried and retrieved using query languages based on captions because the text search is mature and well-implemented in most database systems. A text search can be categorized with the following two types:

1) Search by object identifiers (index or keywords): e.g. Find the images with the filename of The City of Calgary.
2) Search by conditional statements: e.g. Find the images whose cloud coverage is less than 40%.

These two query methods are the simplest ones in large image databases and GIS applications. The first one requires each image to carry one or more keywords (or identifiers). When an identifier is found in the database, the entire image is retrieved. The

second method is the typical data retrieval in relational databases in which a record or several records are retrieved based on the matched properties that may hold for their fields. These two query methods have been successfully implemented in image databases and many GISs.

In GISs, usually there is another query method, called spatial query, that allows users to specify, pick, or point to a spatial location of a graphical display to access spatial objects. For example, one can find how many lakes there are within 100 kilometers of the city of Calgary. The three types of queries and retrievals that have been discussed so far have been successfully implemented in many GISs and image databases. Although useful, they are not good enough for large image queries and retrievals, such as that the users can't use these query methods to retrieve images based on colour, texture and image content.

Therefore, for large image database, two other special query and retrieval methods are required to be implemented:

1) Search by the similarity of objects; e.g. Given an image, find the images that are statistically similar to the given image in the database.
2) Search by the content of a given image;. e.g. Given Lake Louise, find all the images that contain the same object in the database.

These two query methods are more complicated and distinguish large image database from traditional image databases. The first method is performed as described in the last chapter by giving a pictorial pattern (i.e. Lorenz information measure), and asking the system to look for those images that have patterns similar to the given image. The second method, specific for

large image databases, is more complex and requires many difficult image processing and analysis tasks such as filtering, classification and object extraction, etc. In practice, these two query methods can be further divided into the following basic categories as well as various combinations of them [Zhou, 1995]:

1) The retrieval of textual data via a picture. For example: Given an image, show the image quality and image statistical information.
2) The retrieval of images via spatial relationships. For example: Given an image, find images that are to its left and overlap with it.
3) The retrieval of images via sub images. For example: Given an image of a lake, find images which contain the lake.
4) The retrieval of images via similarity. For example: Given an image, find images which are similar to the given image using a given similarity measure.

It is not necessary for all the query methods discussed in this section to be used independently. As a matter of fact, in most cases, they are used together to get better results. For instance, if one tries to find a specific image that contains Lake Louise and the acquisition date is in July, 1985, the best way to do it is firstly to perform Query by Content to retrieve all images that contain Lake Louise. Secondly the sub query should use text query based on the query results of last query to retrieve the image with acquisition date of July, 1985.

4.3 Existing Problems of Large Image Query

In traditional RDBMSs, since the 1970s when SQL was developed by IBM, it has been an accepted standard database query language [ANSI, 1989] and widely used in many modern information systems and image databases. Though it is really powerful for supportint text-based queries and retrievals and spatial data types and their queries support became practical with extended SQL, there are still several problems of using the SQL or extended SQL to handle similarity and content based data queries and retrievals:

1) In traditional RDBMSs, the users access images based on captions (i.e. text). Usually the original keywords do not allow for unanticipated search in subsequent applications. And more important, there are no uniform textual descriptions of image features such as color, shape and content, etc. Without appropriate forms of these descriptions, QBS and QBC are close to impossible.
2) Special large image queries such as QBS and QBC are difficult, if not impossible, to be constructed using only existing relational algebra because of the lack of data processing capabilities of relational algebra. Complicated data processing functions, such as image statistical analysis, classification and feature extraction, etc. have to be integrated into the existing RDBMSs.
3) Extended SQL such as SQL-3 [ANSI, 1993], GIS/SQL [Robinson and Mackay, 1993] will be inefficient in large image handling because of the underlying data model and the lack of effective data management abilities and indexing techniques for large images.
4) While rapid access to the required images is one of the most important functional

requirements of large image databases, access to the required large images can be slow due to massive data volumes of large images and unnecessary relational joins used in many relational databases. That is, large images force the storage of information on disk where access time is much longer than accessing data in memory. Therefore, disk access methods that minimize access frequency have to be found and implemented.

5) Many users have difficulties in using SQL or extended query languages to perform large image queries and retrievals in image databases due to the complicated nature of large images and the lack of navigation and visualization capability in most existing query languages.

6) Complicated queries such as QBS and QBC are difficult to perform without the help of a graphic interface.

Some of these problems and their solutions such as image management methods, indexing techniques and preprocessing techniques have been discussed in the last several chapters. The next section will focus on how the proposed image database implements special large image queries -- Query by Similarity and Query by Content.

4.4 Database Browsing

Browsing is an attempt to break the limitations of querying, particularly for cases where the database is very large and complex and users do not fully understand the contents of the database or when users cannot describe exactly what they are looking for. This situation is very common in large image applications, such as Earth change monitoring, urban planning, as well as in many GIS applications, such as land use, real estate. The browsing techniques

originate with information retrieval and have great potential for applications where a vast amount of information needs to be searched with only imprecise specifications of what is desired. Large image databases are the good candidates for browsing techniques. The ability to browse is generally regarded as one of the strongest reasons for using a visual environment in database querying.

Usually, browsing is implemented through a hierarchy structure, also called is-related-to link type. Figure 4-1 shows a browsing example.

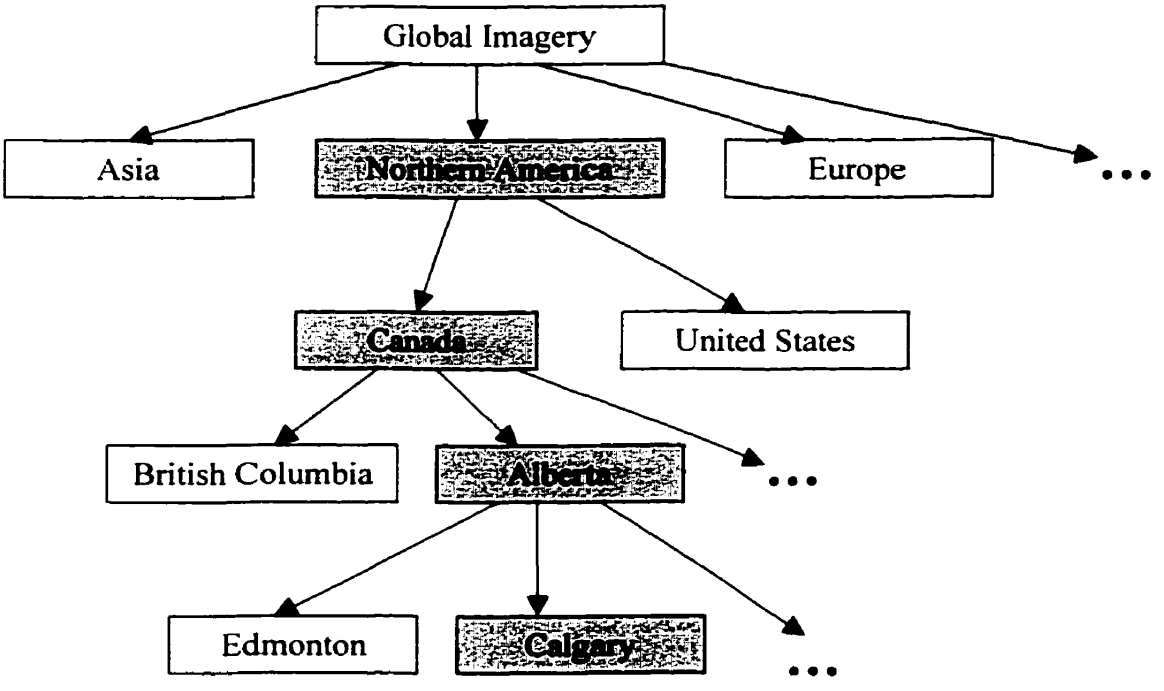


Figure 4-1 An Example of Image Database Browsing

If you want to find a Calgary image, you can start with the image of the world, then browse to Calgary image via a hierarchical path. Browsing through the hierarchy structure is a

delicate process that is constrained on the one hand by the need to have flexible access to the desired information and, on the other hand, by the need to avoid disorientation. In general, the more paths the users are permitted to explore, the more likely the users will eventually get "lost". Thus, the hierarchy structure is used to represent images which have multiple relationships in the database.

The browser for the large image databases should have abilities that allow the users to point and select the areas of interest, to zoom in/out, to pan, to view the object hierarchy, to show the related metadata and attributes, to hide unimportant objects or background images and to walk though interrelationships of images or objects through some windowing ability. For example, in Figure 4-1, the users can select the Northern America area in the global image and zoom in, then go on till the Calgary image is obtained. Browsing can be further facilitated through the linking of computer graphics and query tools.

4.5 Query by Similarity and Query by Content

Traditional RDBMSs usually support only text-based queries. Commercial GISs can provide a limited set of graphic based data query and retrieval capabilities through point and pick, in addition to textual data queries. Compared to these traditional queries, similarity and content based queries and retrievals in large image database have important distinctions:

1) They are approximate and there is usually no exact match. In other words, QBS and QBC techniques serve as "information filters" and simply reduce the search for the users who will ultimately discard false retrievals or visually browse the returned images and select

the ones they want.

2) Visual language and visual interaction are keys to these techniques. A query language is said to be a visual language whenever the semantics of the query are expressed by drawing or picking. Two main advantages of visual languages for database queries are that, firstly, visual languages are more natural than text oriented query languages, and secondly, they allow a combination of operations. Two main disadvantages associated with visual languages are the lack of normalization and expressing negation. These two main disadvantages can be overcome by integration with a text based query language, e.g. SQL. Through this kind of integration, a visual based query language can provide a very powerful query capability to support large images which allows the users to use visual query and visual evaluation and refinement, and to decide what to discard and what to keep.

3) In QBS and QBC applications, through the interaction with the system, the user is offered the possibility of a virtually unlimited set of unanticipated queries rather than having a system automatically classify and recognize samples into a small number of predefined classes. In other words, the main output of these queries is a set of images with desired properties that the users will use for subsequent applications, rather than a limited set of symbolic outputs that are predefined and hardcoded which happens in traditional queries.

To better perform large image queries, a database management system which supports QBS and QBC must meet the following requirements:

1) All images must be analyzed prior to storage so that descriptions of their feature and

content can be extracted and stored in the database together with the original raw images. Therefore, the image analysis work (i.e. the preprocessing tasks described in the last chapter) should be done when the users invoke the read_image( ) function to store the image into database. After read_image( ) is done, the similarity measure and the description of the image content have been calculated and stored in the same database, ready to be queried. The effectiveness of a large image database ultimately depends on the types and correctness of image content representations used, the types of image queries allowed and the efficiency of search techniques implemented.

2) Fast responses are essential to large image databases. A large image database must employ searching methods that are faster than sequential scanning methods, and "scale-up" well (i.e. their performance remains consistently better than the performance of sequential scanning methods as the database grows).

3) Query formulation must be flexible and convenient (as opposed to queries expressed by a command-oriented query language like SQL). Therefore, queries must be specified through a graphical user interface.

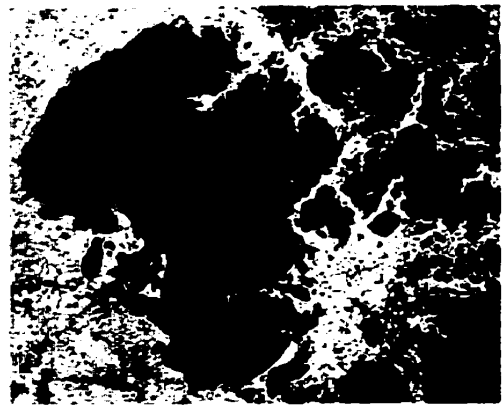
In the proposed visual query environment, the above requirements have been met and implemented.

4.5.1 Query by Similarity

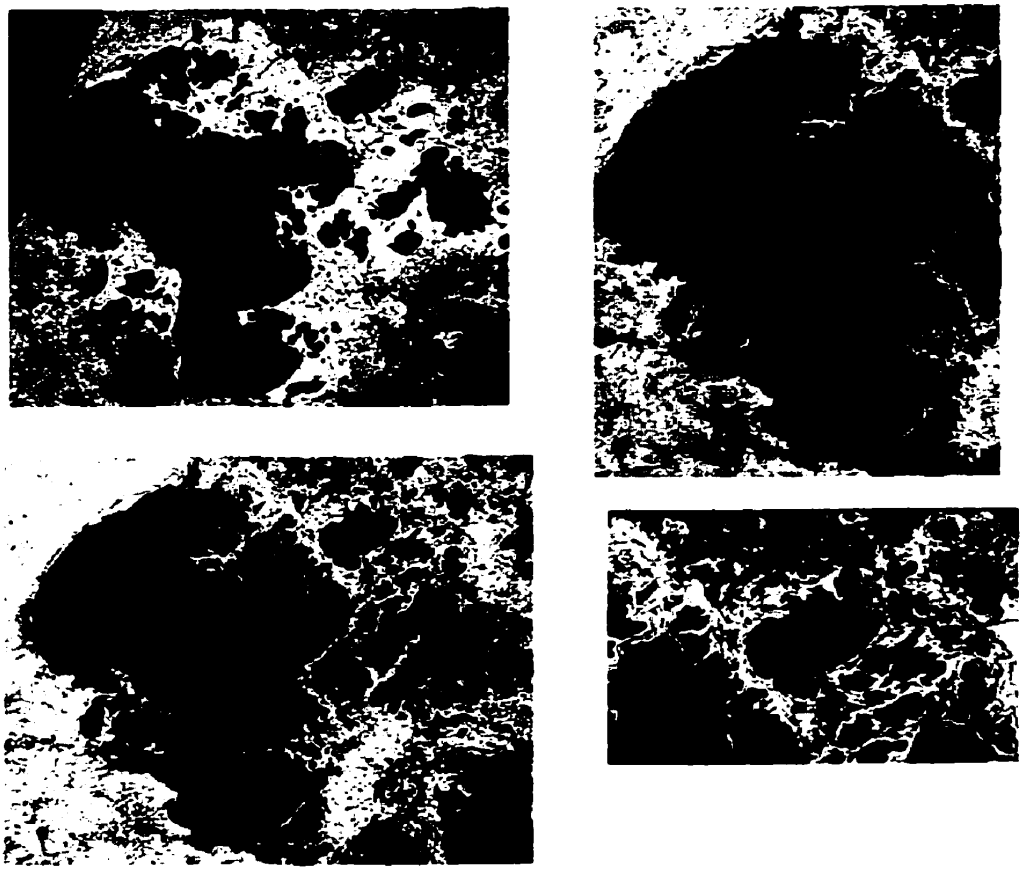
As discussed in the last chapter, when an image is read into the database using the read_image ( ) function, the function calculate_Lorenz ( ) is invoked automatically. Lorenz values of this image are calculated and stored in the corresponding table. The users can issue

a query to retrieve images that are similar to the given image. For example, one can find all images that are similar to the given image (Figure 4-2(a)). The system will then use the calc_similarity( ) method to calculate the similarity measure, which is defined as the summation of the polygonal areas enclosed by the two Lorenz curves as described in Section 3.5, between the queried image and the given image. The principle is that if the similarity measure is below a preset threshold t, the two pictures can be considered as informationally similar in the sense of having similar Lorenz information curves.

Upon successful retrieval, the results are written into a list which an individual object can be selected to perform additional operations such as display, zoom in/out and pan, etc. Figure 4-3 shows an example using this utility. More details about this graphic interface will be discussed in the next chapter. Also, Figure 4-2(b) shows the resulting images that are similar to the given image shown in Figure 4-2(a).



(a)



(b)

Figure 4-2 Example of Query by Similarity:

(a) Original Image (b) Some Resulting Images

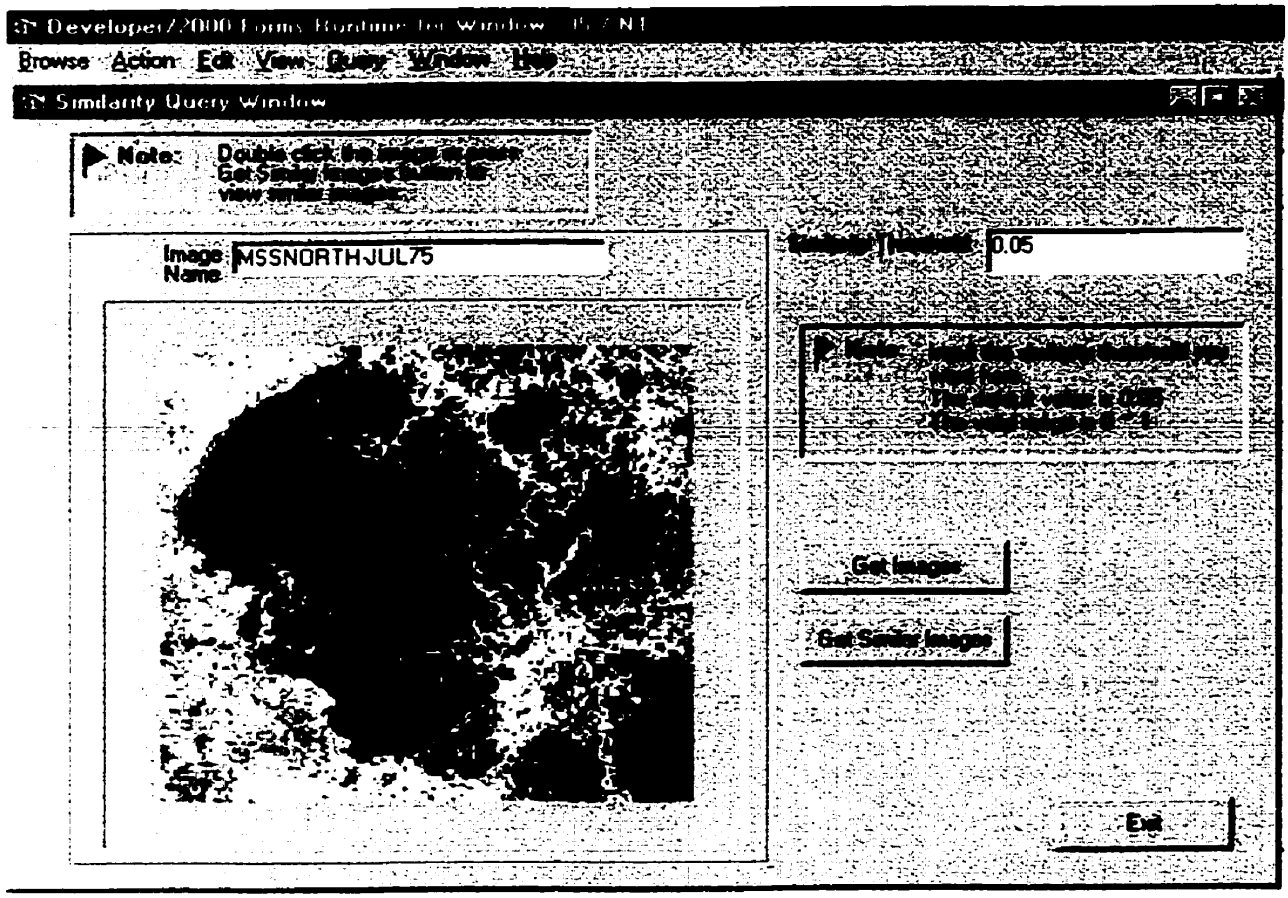


Figure 4-3 A Graphic Interface Used to Do Image Operations

It is necessary to note that, because the Lorenz information curves of all images are always normalized, curves for pictures having different sizes can be plotted and compared. This is critical and important to multiresolution image databases and multiresolution image applications.

4.5.2 Query by Content

Also, as was discussed before, when an image is read into the database using the read_image ( ) function, the function content_extract ( ) is invoked automatically. Then the Content

Relationship Graph (CRG), discussed in the last chapter, for each image is computed and stored in the corresponding table. It has to be noted that the "content" mentioned in this research is limited to parameters that are feasible to compute given the state of the art in computer vision today, such as shape and layout, etc. It is not, for the moment, attempted to automatically derive more complex semantic descriptions such as "building", "city" etc., which are currently beyond the reach of object extraction technology. These descriptors should be entered manually, if necessary.

Content based queries are then performed by the users through constructing a query specifying the contents to retrieve. For example, given Lake Louise, find all images in the database that contain Lake Louise. With CRG, the problem of retrieving images which contain the given content is transformed into a problem of searching a database of stored CRGs. This technique can be illustrated using Figure 4-4. Figure 4-4 (a) shows a given object and its content description. Let $Q = (q1,q2,q3,q4,q5,q6)^T$ be a vector of properties of the given object, where q1,q2,q3,q4,q5,q6 represent area, perimeter, centroid (i.e. geometrical center), roundness, distance and the orientation of the given object respectively. Let $S=(s1,s2,s3,s4,s5,s6)^T$ be a vector of properties of the compared object. Then the distance between objects Q and S is defined as the minimum distance computed over all possible objects:

$$Dist\ (Q,S) = \left[\sum_{i=1}^{K} w_i \mid q_i - s_i \mid^p \right]^{1/p} \qquad (4\text{-}1)$$

where p is the order of the metric, K is the total number of content measures, w_i is the normalized weight for each parameter. They are set to the following default values:

$w_{area}=0.2$, $w_{perimeter}=0.15$, $w_{masscenter}=0.25$, $w_{roundness}=0.2$, $w_{orientation}=0.1$, $w_{distances}=0.1$

These weights can be changed by the users. The proposed system has to make the weights qualify the following condition:

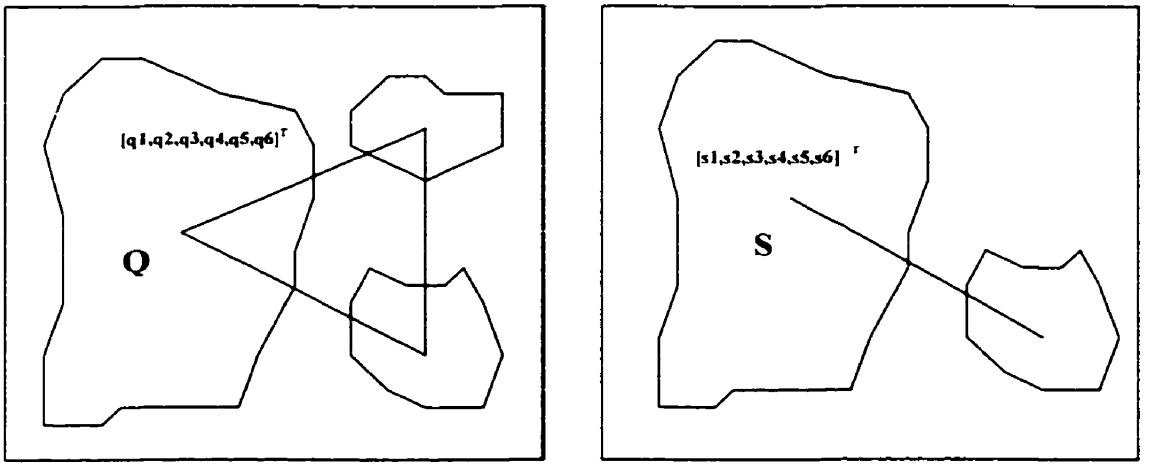
$$\sum_i w_i = 1 \qquad \text{where} \quad w_i = \frac{\omega_i}{\sum_j \omega_j} \tag{4-2}$$

where ω_j are the weights input by the users.

For p = 1 and p = 2, the Manhattan and the Euclidean distances are obtained respectively. QBC requires that all objects within distance t must be retrieved. Specifically, all images that contain the object which qualify the following condition have to be retrieved:

$$Dist\,(Q, S) \leq t \tag{4-3}$$

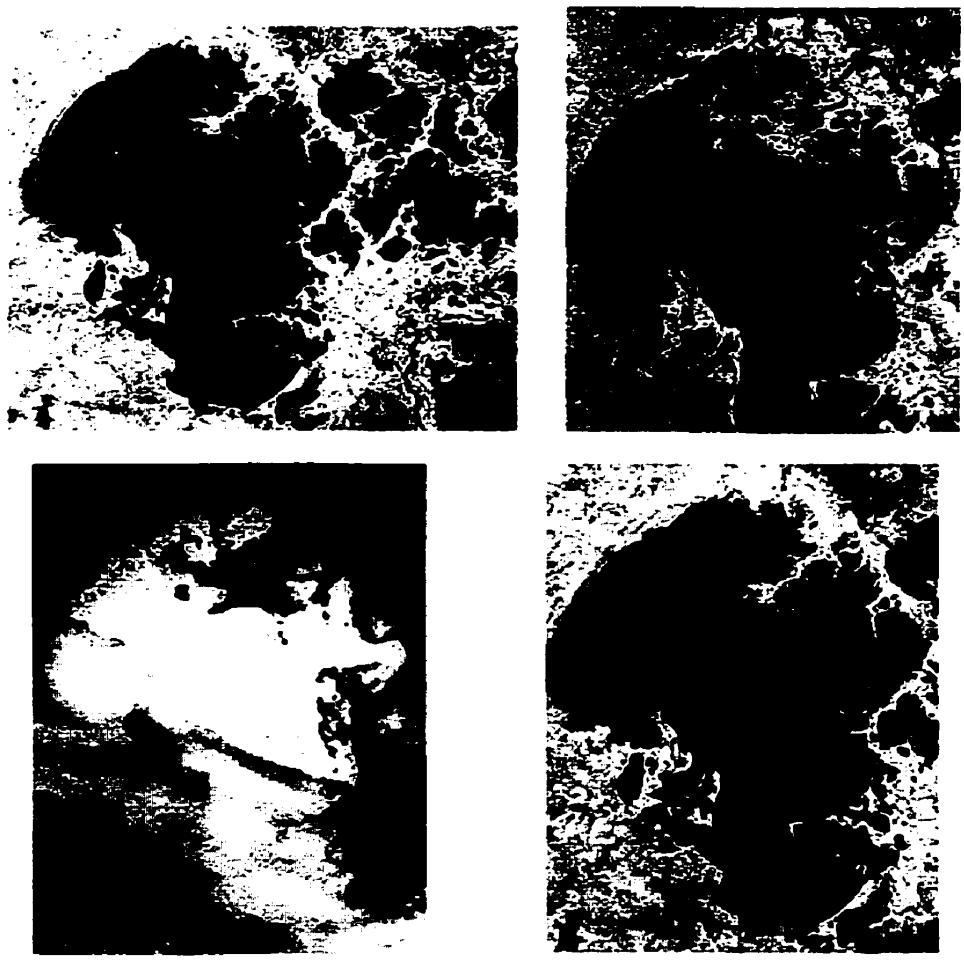
Without loss of generality, p = 2 is used. However, the proposed method can handle any order of metric. Figure 4-5 shows the given object and the resulting images of QBC that contain the same object as the given object.



(a) **(b)**

Figure 4-4 QBC Technique Using CRGs
(a) CRG of Given Object (b) CRG of Queried Image

(a)



(b)

Figure 4-5 Example of Query by Content:

(a) Original Image, (b)Some Resulting Images

It should be noted that in the proposed system, the Lorenz information measure and image content of most images are calculated, extracted and stored in the database when the read_image( ) function is called before any query is invoked. Other than that, the proposed system also supports dynamic information measure and image content calculation and extraction. For example, given a new image that is not stored in the database, find all images that are similar to the given image. In this case, the Lorenz information measure of the new image will be calculated before $d(f,g)$, the similarity measure of two images f and g, is calculated.

4.6 Discussion and Limitations of QBS and QBC

4.6.1 Discussion of Multiresolution Issue of QBS and QBC

Since most large images that need to be handled in our large image database are remotely sensed images and there are different types of remotely sensed images from different remote sensors used to collect and record specific types of energy of different wavelengths, it is important to address here how to use the visual query methods presented in the previous sections appropriately to retrieve the remotely sensed images of potential interest of the users.

Multiresolution is one of important characteristics of remotely sensed images. The terminology multiresolution in remote sensing area has three-fold meanings:

- Multiscale: the data from different remote sensors have different ground resolution. For example, TM generates 30m spatial resolution data and 120m data for infrared band,

MSS 79m resolution data, SPOT 20m data and 10m data for panchromatic band. Some military satellites can even generate data with 1m spatial resolution.

- Multispectral: the data have multiple bands which cover different spectral ranges as discussed before.
- Multitemporal: the data are collected in different seasons or epochs.

While multiresolution data provide benefits for different applications, however, they imply problems in remotely sensed image query and retrieval.

4.6.1.1 Multiresolution Issue of QBS

There are several remote sensors collecting and recording energy from the objects on the Earth in different wavelength ranges. For example, the spectral range of MSS (Multispectral Scanner) and TM (Thematic Mapper) systems includes visible, near-infrared and thermal infrared; the information from each narrow wavelength range is gathered and stored in a channel, also sometimes referred to as a band.

Table 4-1 summarizes the spectral sensitivity and spatial resolution of each of these systems. While thermal sensors use light detectors sensitive to the direct contact of photons on their surface, to detect emitted thermal radiation so that they essentially measure the surface temperature and thermal properties of targets; radar sensors detect and record the microwave energy of the target objects. In other words, the data from different remote sensors have different spectral resolution. As discussed before, the algorithm of QBS is based on Lorenz information measure, i.e., based on gray level value distribution of the images. Therefore, the

QBS method is spectral resolution dependent. In other words, query comparison of QBS should be based on same spectral resolution, i.e., same bands. Otherwise, the retrieval results of QBS and QBC would be meaningless.

Sensor	Mission	Sensitivity (um)	Nominal spectral location	Resolution (m)
MSS (Bands: 1-4)	1-5	0.5 - 0.6	Green	79/82*
		0.6 - 0.7	Red	79/82
		0.7 - 0.8	Near-infrared	79/82
		0.8 - 1.1	Near-infrared	79/82
TM (Bands: 1-6)	4,5	0.45 - 0.52	Blue	30
		0.52 - 0.60	Green	30
		0.63 - 0.69	Red	30
		0.76 - 0.90	Near-infrared	30
		1.55 - 1.75	Mid-infrared	30
		10.4 - 12.5	Thermal infrared	120
		2.08 - 2.35	Mid-infrared	30

* 79 m for MSS-1 to MSS-3, and 82m for MSS-4 and MSS-5.

Table 4-1 Sensors Used on MSS and TM

From this point of view, QBS has advantage in handling multitemporal images because it is not hard to differentiate the images collected in different seasons by using this method. For example, given an example image that covers Alberta collected in July 1990, one can find other images that cover the same area and are acquired in the same season by using QBS.

It should be noted that in TM images, there is an exceptional Band 6, the thermal infrared band. Its pixel values related to the surface temperature and thermal properties of targets. Therefore, it cannot be compared to other bands of TM and MSS images.

The multiscale issue doesn't greatly affect QBS because Lorenz information measures are always normalized as mentioned in Section 3.4, images having different spatial resolutions

can be compared.

4.6.1.2 Multiresolution Issue of QBC

As discussed before, the algorithm of QBC is based on the 6 property parameters of object polygons that are extracted from raw images. They are area, perimeter, centroid, roundness, distances and orientation. These polygon property parameters describe the size, shape and position of the polygon, i.e. they are size, shape, position dependent parameters. Therefore, in most cases, they are considered to be spectral resolution invariant while they are spatial resolution and acquisition date dependent. That means when spatial resolution of an image changes, such as a TM image with 30 pixel resolution is resampled into 60m, 120m, 1000m pixel resolution, the area, perimeter, distances of the 6 parameters mentioned above will change. Why they change and how to quantatively describe this change are beyond the scope of this research. Many researchers are still working to address this problem. Also, acquisition date difference can greatly affect the 6 parameters. For example, in Figure 4-6, the same lake collected in different seasons shows different shapes. Therefore, when QBC method is used to query multitemporal images, the edge extraction results and the selection of threshold t are more important than other situations. With accurate edge extraction results and appropriate threshold section, the query and retrieval results of multitemporal images can be greatly improved. We'll see the testing results of multiresolution image query and retrieval in the Chapter 5.

(a) (b)

Figure 4-6 Baril Lake Collected in Different Seasons (a) June 1979, (b) May, 1974

In this research, in order to avoid the problems multiresolution data cause, it is strongly suggested that each remotely sensed image should be radiometrically and geometrically corrected before it is stored in the database. Then the 6 property parameters are calculated based on geographic coordinates of polygons instead of pixel coordinates and pixel resolution of images. There are two main advantages to calculate the 6 property parameters in this way: one is that they are scale and translation invariant; the other is that they are standardized and can be used for comparison because it is easy to make all images registered in the common coordinate systems.

In some cases, such as when some large image databases include lots of historic data which means that they might not be properly registered, georeference information is not available. Thus two property parameters, perimeter and distance, are suggested not to be included in the calculation of polygon comparisons because these two parameters are ground resolution dependent. Therefore, it is important that the visual query interface has the ability to choose which parameters should be used to do content based queries and retrievals.

It should be noted that even the method described above cannot handle some extremely hard situations. For example, if Lake Louise in an image is resampled into just a few pixels, the shape of the lake has been lost and the 6 property parameters calculation using the methods discussed above would be greatly inaccurate.

4.6.2 Limitations

Similarity based and content based queries are part of the most difficult problems in large image database. Lots of work is currently under way to address these problems. There is no method that can be used as "panacea" to solve these problems in every situation. Therefore, there are limitations in the proposed QBS and QBC methods.

In this research, QBS implementation is based on the Lorenz information measure. Because the Lorenz information curves are always normalized, images with different spatial resolution can be compared noting that images derived from different sensors are not recommended to be compared as discussed before.

For QBC, there are several limitations:

1) Non-polygon objects: QBC is not appropriate for objects retrieval such as river, vegetation area and road, etc, called non-polygon objects because these objects cannot be represented using closed polygons based on which QBC is implemented.
2) Boundary problem: If an object resides across the boundary of the images, as shown in Figure 4-7 (lakes A and B extend across the boundary), the closed polygon cannot be obtained. Therefore, QBC is not likely to be useful in this situation. A method can be

used to solve this problem is to, firstly, mosaic this image and its neighbor image together to form an image with a complete object, then secondly extract the closed polygon of this object.

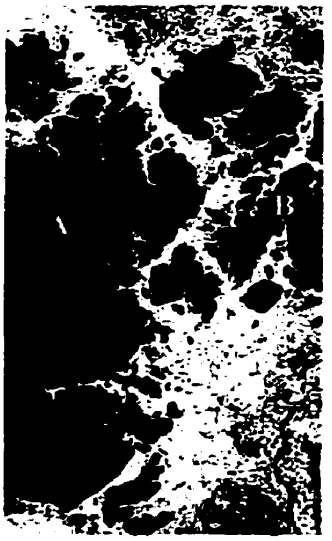


Figure 4-7 An Image with Two Lakes Extending Across the Boundary

3) Multiresolution problem: as discussed in the last section.

4) Uncertainty problem: As Alesheikh [1998] described in his thesis, uncertainty, including measurement, model, processing and transformation, data usage uncertainty will affect the accuracy of the 6 parameters that are used to describe the object polygon [Alesheikh, 1998]. That is, the uncertainty will affect the results of QBC. In some cases, the effect will be significant. Therefore, how to handle these uncertainties in large image databases will have to be studied in future work.

4.7 Summary

This chapter focused on query and retrieval methods of large image databases. Firstly,

several possible types of large image queries and retrievals were listed and discussed. Secondly the complexity and difficulty of these special large image query methods were highlighted. Thirdly, the database browsing technique was discussed. Fourthly, characteristics of visual based query and requirements for a system that can support QBS and QBC were discussed. Finally, how to perform Query by Similarity and Query by Content were discussed in detail. These techniques have significantly enhanced the capability of traditional query languages, e.g. SQL, and made the proposed large image database much more effective, efficient and easier to use. Also, some notes on how to appropriately use QBS and QBC were included. Multiresolution issues and limitations of QBS and QBC were finally discussed.

CHAPTER 5 A PROTOTYPE LARGE IMAGE DATABASE DEVELOPMENT

5.1 Introduction

In this research, the main purpose is to study how to effectively and efficiently store, manipulate, query and retrieve large images in relational databases. After the large image management, indexing methods and query methods were discussed in the last several chapters. This chapter will focus on the design and prototyping of the proposed large image database. The system is supposed to effectively demonstrate the concepts and techniques presented in this research. Thus, the important considerations in the design and implementation of the proposed system are:

1) Powerful management, indexing, processing and query abilities. That is, the proposed system should support advanced management, spatial access methods, image processing and special image query functions, such as QBS and QBC, in addition to traditional RDBMS capabilities.
2) User-friendly graphic interface. This is important to perform visual based query and query refinement.
3) Ease of use. This is extremely important due to complexity, large data volumes and the rich semantics of large images.
4) Extensibility. The proposed system should allow users to extend it to include more powerful capabilities or tailor the system to their application specific requirements.

The rest of this chapter is organized as follows. Section 2 discusses the overall system

architecture, software components and general implementation considerations of the proposed large image database; Section 3 focuses on the development and implementation of the Graphic User Interface; Section 4 shows the test and analysis results on the system's usability, capabilities and performance using a small test image database which mainly consists of satellite images; Based on the discussion of Section 4, the advantages, disadvantages and limitations of the proposed system and proposed query techniques are provided in Section 5 before this chapter is concluded.

5.2 Overall System Architecture

The proposed large image database system consists of the following components:

1) Graphic User Interface (GUI)
2) Query Processor
3) Relational Database Management System (RDBMS)
4) Image Database
5) Processing function libraries

Figure 5-1 shows these components and their inter-relationships. The GUI provides a user-friendly visual environment for the users to browse images in the database when they do not fully understand the contents of the database or when they cannot describe what they are looking for. It also allows the users to manage database, edit data and perform query without the knowledge of SQL syntax, perform operations such as zoom in/out and pan the images and perform visual queries -- QBS and QBC in an easy and understandable way; the Query

Processor will decide, when a query is issued, if the query is logically valid, what kind of query it is, calls the corresponding functions to perform query calculation and activates the database retrievals; the RDBMS provides data management abilities and ensures data integrity, security and manage user requests; the processing function libraries provide abilities to perform image preprocessing and other image and GIS operations; the image database stores the large images and their related information using a relational database.

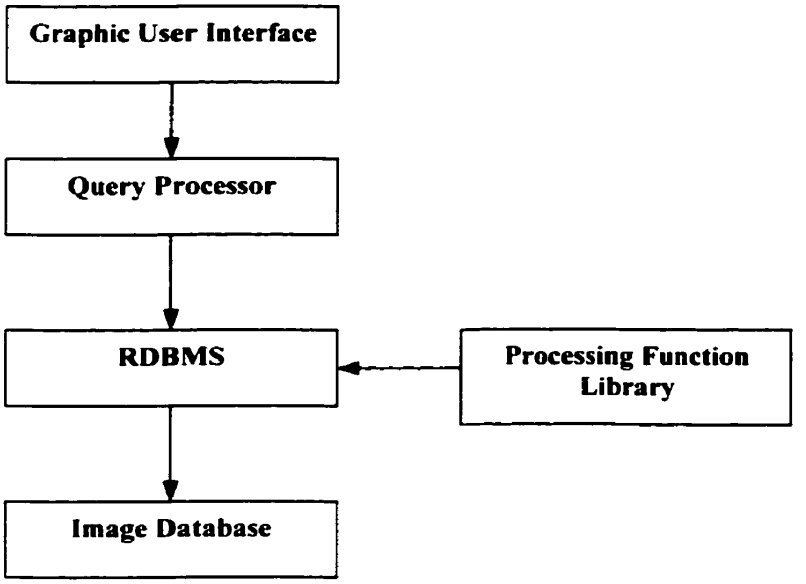


Figure 5-1 Architecture of the Proposed System

Considering that the proposed system is supposed to support multiuser applications, the security of the image database is the big issue in terms of database integrity. That is, the access to image database is limited. Only those users that are granted the privileges can access to the image database to update, delete and insert data into the tables. Other users without access privileges will work under the user account. When the users log on to the user account, they just have privilege to query the data that are stored in the image database, but

they don't have privilege to update or insert new data. If they want to change the data, for example, they are not satisfied with the classification results of a certain image and want to use their own image classification results, they can do image classification using the functions provided by the function library, store the results in their own tables and do queries and retrievals they want. If their new classification results are good and they want to store them into the image database for future use, they have to ask the database administrator to do it or they can do that with being granted the corresponding privileges. In this way, the users work with a working copy of the image database, i.e. temporary tables. To the users, there are no differences from working on base tables that store image data and related metadata; but the database integrity is significantly improved.

Under multiuser situations, there is another issue that is needed to be considered when designing the database. One image may have different preprocessing versions created by different users in different applications. They are all supposed to be kept into the image database. Therefore, the proposed database should handle multiversion issues. In this research, there is one field in the image metadata table, called version_no, that stores version numbers of different image preprocessing results. Every time a new preprocessing result of the existing image is inserted to the table, the version_no will be incremented by 1 automatically by invoking the attached trigger.

The proposed system is a complicated system and it is not feasible for this research to develop the whole system from the beginning. Fortunately, there are many commercial database management systems that provide the various parts and powerful functionality that

are necessary and important to the proposed system. It's more practical to use these systems as a basis on which to build the proposed system. To name a few, these commercial RDBMS include Oracle of Oracle Corp., Paradox of Inprise Corp., SQL Server of Microsoft Corp. and Sybase of Sybase Inc., etc. Oracle8™ was selected as the basic system for the prototype system to be built upon because [Oracle, 1998]:

1) It is the most popular and most powerful DBMS in the world. Thus it is a good candidate to be chosen in this research as a large image DBMS.
2) It supports high-performance transaction processing and large user populations. Since the huge amount of data of large image databases usually significantly declines the performance, especially the speed of data access, the high performance technology is one of the most critical issues in large image databases.
3) It supports reliable query results. Oracle8™ effectively supports mixed workload environments characterized by simultaneous query and update activity. Many databases force users to choose between good performance and guaranteed data consistency, one coming at the expense of the other. Oracle8™'s multiversion read consistency always provides users with consistent query results, while never imposing a performance penalty on concurrent update activity.
4) It provides advanced techniques to support very large databases. This is one of the most important reasons that Oracle8™ is employed as based DBMS in this research. Oracle8™ addresses the largest and most demanding data warehousing applications with databases into the terabytes and beyond by providing partitioning, rich query processing techniques, sophisticated SQL optimizer and extended backup/recovery techniques, etc.

5) It provides a Spatial Cartridge to support spatial information management. While few commercial DBMSs provide spatial information support or they need to integrate the third-party products to meet this requirement, Oracle8™ Spatial Cartridge allows the users to efficiently store, access, and manipulate their spatial data in the same way as the users' structured data.

6) It supports Web applications. It can be used to fully integrate the users' existing Oracle8™ business applications with Web technology and safely deploy them inside or outside the corporate firewall. Oracle8™ is the only open systems solution that meets the demands of high-transaction on-line transaction processing (OLTP) systems, query processing in large-scale data warehouses, and manageability requirements of businesses that are distributing data throughout the corporate enterprise and the Internet. This is very important for future work of this research because finally the large image database will be required to meet the requirements of on-line databases.

As was discussed before, a user-friendly and powerful GUI is important and critical for large image database applications and visual query language implementation. There are lots of GUI development tools that can be used to implement the GUI of this database, such as Visual Basic, Visual C++ and Tcl/TK, etc. In this research, Oracle Developer/2000 is employed as the GUI because of the following features[Oracle, 1998]:

1) It fully integrates with Oracle8™. The users can quickly and easily build robust, enterprise-class applications that enable end users to retrieve, enter, modify, and save information in a database by typing information into on-screen forms.

2) It provides a user-friendly, powerful, objected-oriented development interface.

3) It provides array operations and stored procedures. With Oracle Developer the users can perform array updates, inserts, and deletes and fetch multiple rows at once during queries. This greatly reduces network traffic, which can mean significant performance increases, especially for users running on wide area networks. The user can also perform data manipulation from within stored procedures, enabling sets of updates, including updates to multiple tables, to be sent to the server in a single round trip. Again, this reduces network traffic and improves performance.

4) It provides sophisticated web-based capabilities in reports. The users can easily incorporate hyperlinks, tables of contents, bookmarks, animation, and interaction into their reports. These features help make end users productive, because they can navigate quickly to the information they need, and they can understand it quickly.

5.3 Graphic User Interface Implementation

In Oracle8™, SQL*Plus is the only interface through which the users can query, update, edit and manipulate data that stored in the database. SQL* Plus is a text, command oriented interface which can't handle image display or manipulation. The proposed system is supposed to support image display, editing and visual queries, etc . which must need a graphic environment. From this point of view, the GUI for the proposed system should meet the following requirements:

1) It should be user-friendly, easy-to-use.

2) It should provide basic abilities for the user to browse the data, query, edit and

manipulate data without spending time to learn SQL syntax.

3) It should offer abilities to perform the basic image operations, such as zoom in/out, pan and resample, etc.

4) It should support non-exact and visual queries and retrievals which require the system to graphically display queried images, list query results to allow the users to refine them or subquery them.

In this research, Oracle Developer/2000 is employed to develop the GUI. The GUI mainly consists of the following components: Main Window, General Query Window, Visual Query Window and Processing Function Libraries.

5.3.1 Main Window

Figure 5-2 graphically shows the Main Window. On its menu bar, it has a menu list which includes Browse, Edit, View, Action, Query, Window and Help items which will pop-up new menu items for selection when activated. Figure 5-3 shows these pop-up menus, which have the following abilities:

Browse pop-up menu: Under this menu, the users can browse each image that is stored in the database by selecting the "All images" item, or browse based on the type of the images by selecting Satellite Images, Scanned Images and so forth. The types of images that can be browsed are predefined by a database designer.

Action pop-up menu: Under this menu, there are items Clear all, Save, Print, Exit to clear data, save data into database, print data to printer in current focused item and exit from

current active window.

Edit pop-up menu: Under this menu, there are items Cut, Copy, Paste and Edit to provide standard database editing commands.

View pop-up menu: Under this menu, there are items Fit Window to adjust the displayed image to fit the window size, Select Region to select the region on which the users want to operate, Zoom in/Zoom out to zoom in/out the displayed image, Pan to pan the displayed image and Scale Factor to allow the users to define the image size viewed in the image preview window.

Query pop-up menu: Under this menu, there are items General Query, Query by Similarity, Query by Content to pop up a general query, that is, text-based query window, QBS window and QBC window.

Window pop-up menu: Under this menu, there are items Cascade, Tile and Arrange Icons to provide standard operations to adjust the opened windows positions.

Help pop-up menu: Under this menu, there are items Help, Keys, List, display Error to provide on-line help information, short cut key list and display error number and error message of the run-time errors.

After a query searching is done, the query results will be listed in the Resulting Images area. The users can double click the image name to preview images and image locations or the person names who hold the image will appear in the "Location" area. In this way, users can easily refine their query results through further selecting/viewing of those retrieved results

to decide which image they want. At the same time, the metadata or the attributes, of the displayed image will appear in Attributes text box. The users can edit or update the metadata if they have privileges.

5.3.2 General Query Window

In the menu of the Main Window, the General Query Window can be invoked by selecting the Query menu, General Query item. The General Query Window is a graphic interface through which the users can issue any text-based query without knowing any SQL syntax. Figure 5-4 shows this window. In the menu of this window, there are items Action, Edit, Query to provide basic database operation such as transaction commit, transaction roll back, data saving, copy, paste and query execution, etc., Block, Field, Record to provide functions to navigate the data stored in the table based on block, field or record level.

Also, under the menu, several groups of tool bars are implemented to provide basic database operations and database navigation abilities. At the lower part of this window, several most often used operations and navigation buttons are also provided for users to perform these operations handily.

Consider a query example on how easy it is to perform a text-based query through this window. Assuming the users want to find all TM images that are acquired between May, 1975 and May, 1980, type "TM*" in the "Image Name" field and ">5/01/1975 and <5/31/1980" Then press the Query button, and the resulting image will be listed in this Window. The users can insert, delete, update and edit records in this window if they are

granted privileges.

5.3.3 Visual Query Windows

Visual Query includes Query by Similarity and Query by Content. The QBS Window and the QBC Window can be invoked by selecting the Query by Similarity item and Query by Content item under Query item of Main Window menu. Figure 5-5 shows the Query by Similarity Window. In the image display window, the users can browse the images that are stored in the database to choose the example image. Once the example image is chosen, set the threshold value for similarity measure comparison in the "Threshold" item, then press the Get Similar Images button or double click the displayed image. The resulting images that are informationally similar to the given image will be listed in the "Resulting Images" area of the Main Window. It's quite easy for the users to quickly scan and preview the resulting images in "Preview" window to discard the images they don't want.

Figure 5-6 shows the Query by Content Window. It is quite similar to the Query by Similarity Window except for two items. One is the example object item. When the users double click the displayed image, the objects that are contained in this image will be listed in the "Objects" area and the users can choose which object they want to use as the example object. The others are "Content Measures" text boxes. The users are asked to input the weights for each parameter. If one text box is left blank, this parameter won't be included in the QBC query comparison. In this way, the users can decide which content measures they want to have included into the object comparison calculation and can set up the weight values they want to be used for each parameter because, in some cases such as for images

that don't have georeference attributes, not all the content measures are available to be included into the calculation. This requires that the users have knowledge and understanding of remotely sensed images and their applications. That's the reason the knowledge manager or rule manager should be integrated into the proposed system in the future work to help the users to perform image queries more effectively and efficiently. Chapter 6 will discuss this issue in details.

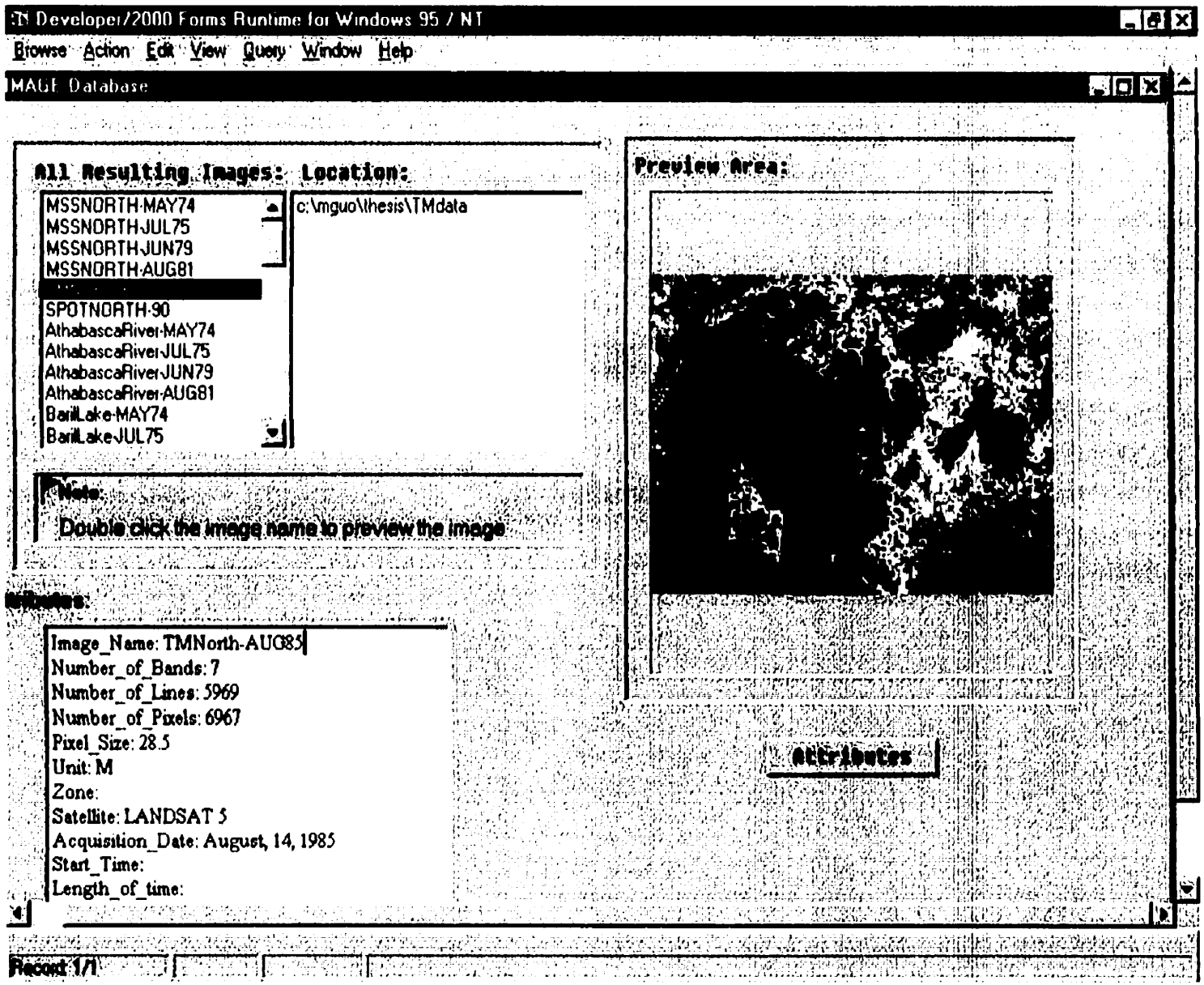


Figure 5-2 Main Window of Developed GUI

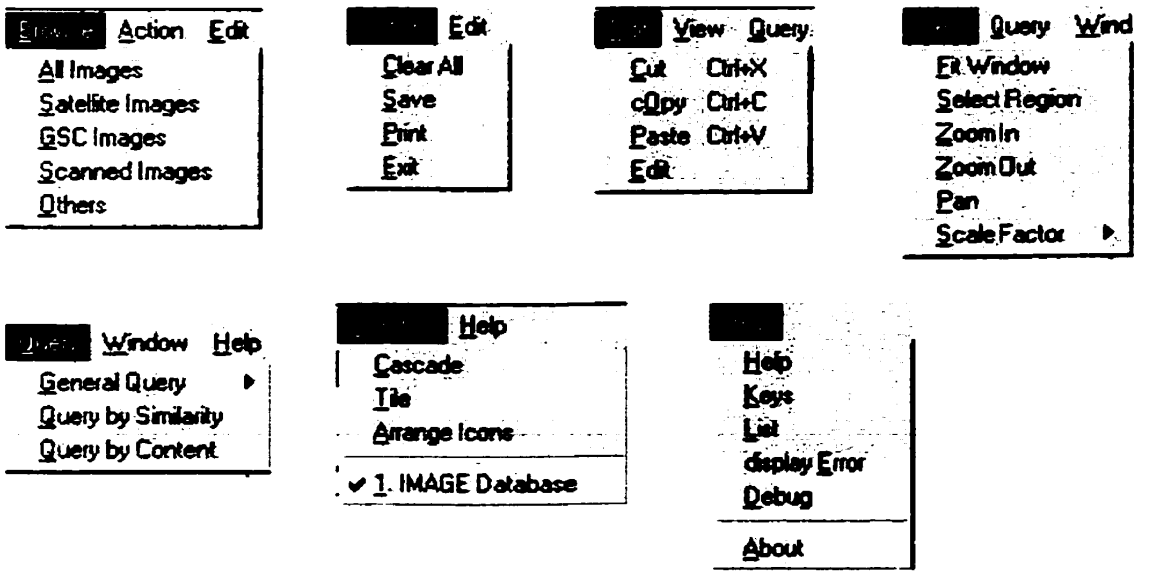


Figure 5-3 Pop-up Menus of the Main Window

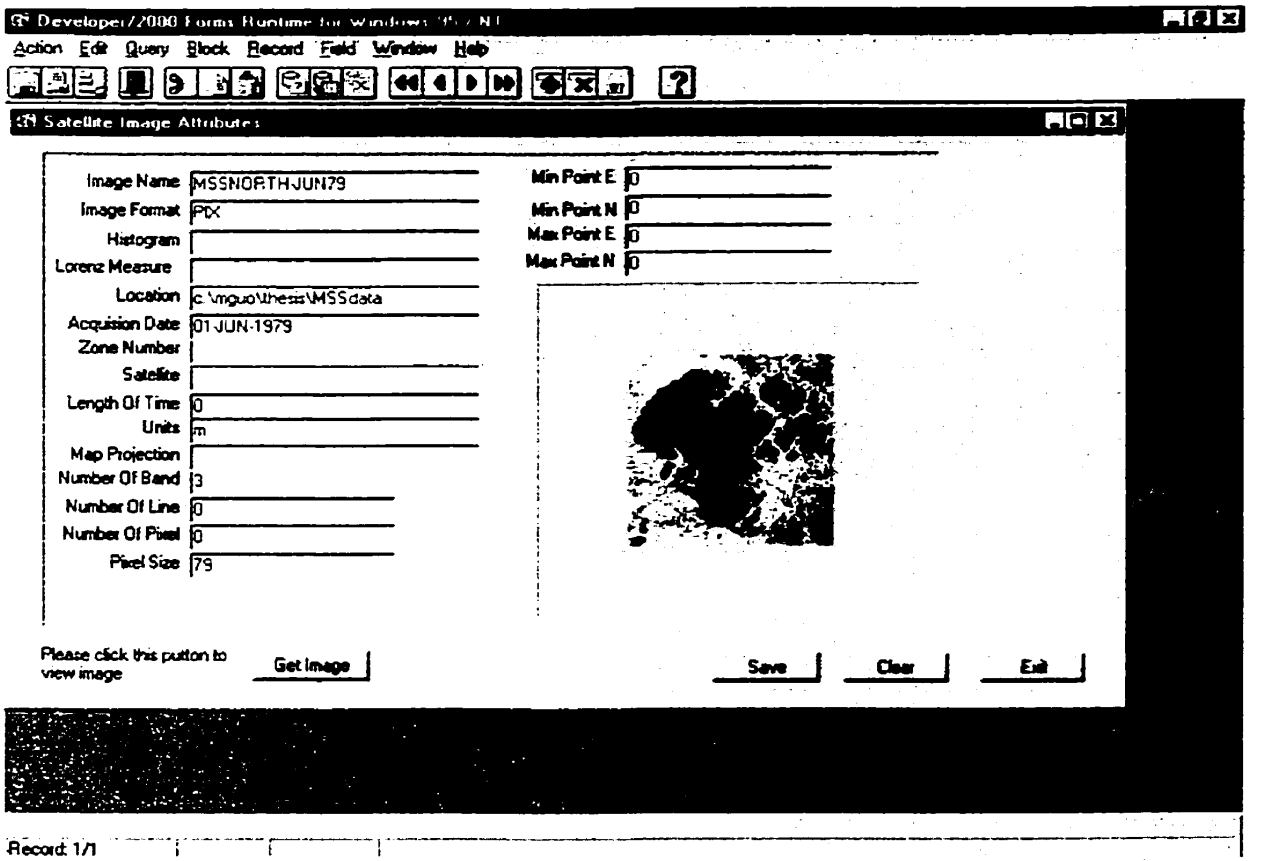


Figure 5-4 General Query Window

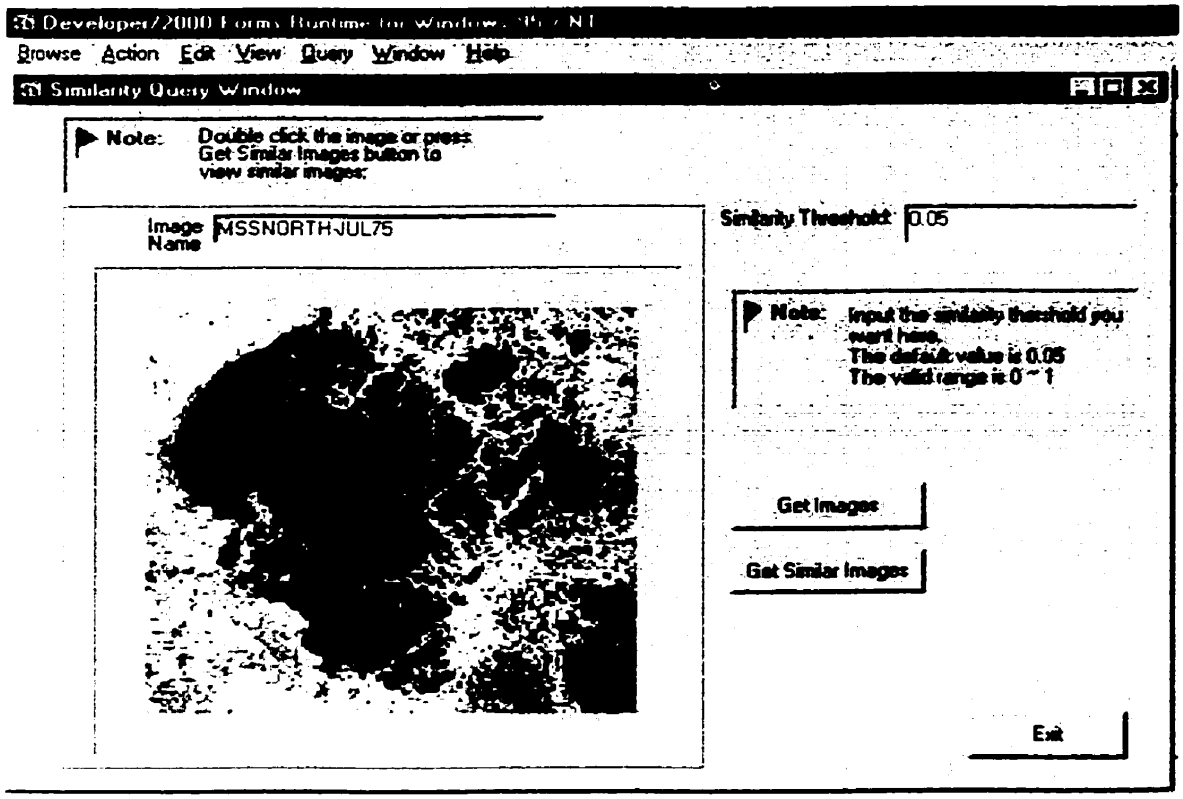


Figure 5-5 Query by Similarity Window

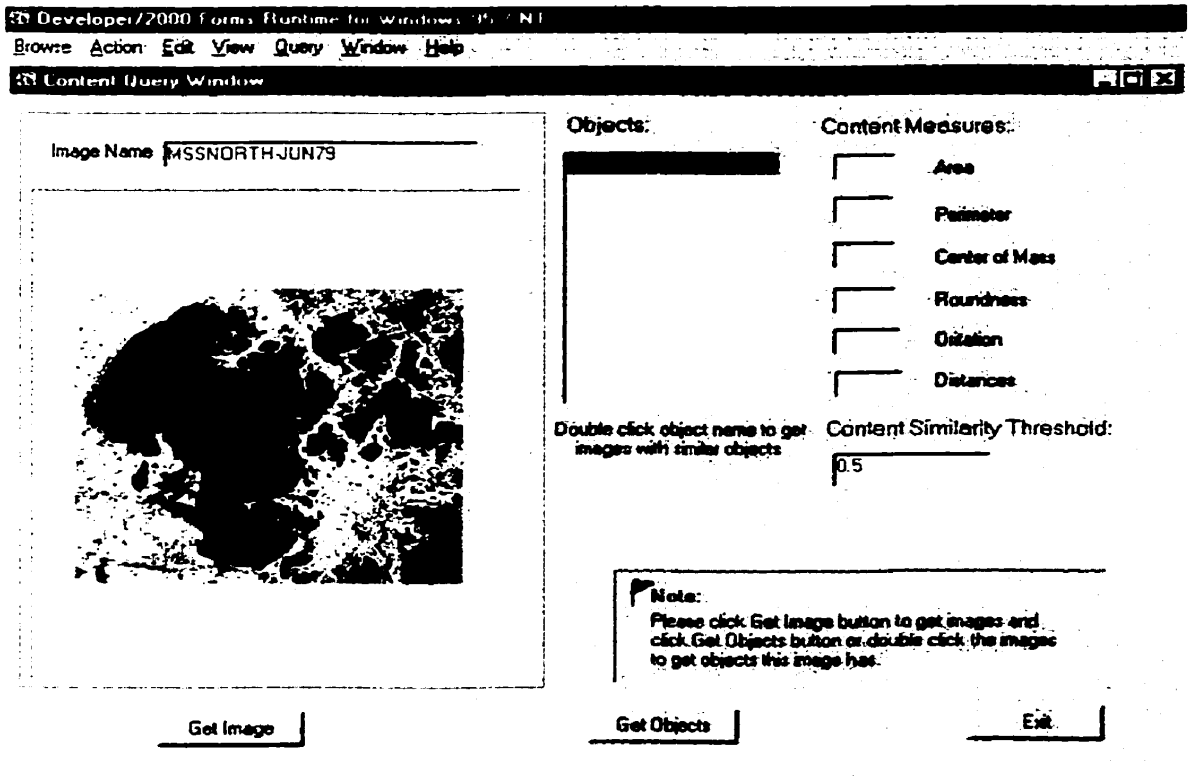


Figure 5-6 Query by Content Window

5.3.4 Processing Function Libraries

Image processing functions, such as reading an image, writing an image in a specific format, classification and histogram calculation, etc., query processing functions, such as Lorenz calculation and image content calculation are implemented using PL/SQL, extension of industry-standard SQL, which is Oracle's powerful and flexible procedural language and is tightly integrated with the Oracle database server. These functions can be classified into the following three groups:

1) Standard functions perform Developer 2000 operations other than built-in functions that are provided by Developer 2000. They perform basic database functions such as execute query, save query, save query results, create R-tree and insert code into R-tree, etc. Standard functions also include image display and operation functions such as display image, zoom in/zoom out and pan, etc.
2) Image processing functions include image compression, filtering, classification, edge detection and linking and content extract, etc., some of which are automatically called when necessary.
3) Similarity calculation and content calculation functions include the Lorenz information measure calculation, image content calculation based on image classification and content extraction.

When an image is read into the database by calling the read_image( ), some functions such as classification, Lorenz calculation, image content extraction and calculation will be invoked automatically to calculate similarity and content of the image and store the results in

corresponding tables. Human interference is allowed during image reading if the edge detection or content calculation results are not satisfactory by manually editing the automatic processing results. When a query, such as query by similarity, is invoked, cal_similarity( ) function will be automatically called to calculate the similarity difference between the given images and other images in the database.

In Developer 2000, it's easy to add a new function to the function libraries using built-in PL/SQL editor. Also the functions, called foreign functions, implemented by other 3GL programming languages, such as C/C++, Fortran, Cobol and Pascal, etc. are easy to be integrated into the proposed system because Developer 2000 provides a powerful interface to invoke these foreign functions within PL/SQL code. In this way, reuse of existing code is possible and can significantly enhance performance or provide additional functionality to the proposed system.

5.4 Tests and Analysis of Test Results

As described in the research background section, the proposed system is supposed to support environmental applications as CCEDA requires. Therefore, it will deal with a large number of complicated images, most of which are remotely sensed images, and complex natural patterns. Because of its complexity, it needs to be extensively tested using carefully designed experiments to evaluate its performance and query and retrieval effectiveness.

The test database in this research consists of 128 multispectral, multitemporal, and multiresolution remotely sensed images, 18 scanned images and their related metadata (text

data and graphic data). Among remotely sensed images, there are 98 MSS images, 6 TM images and 5 SPOT images. All of them cover the same area, North Eastern Alberta. Table 5-1 shows the acquisition dates and pixel sizes of the testing remotely sensed images. It should be noted that the test database is a simulated database because no real database is available for the testing. Neither is the test database meant to be complete, nor can it be treated as a real world application. Real world applications will be, without any doubt, much more complicated than this test database. The goal here is only to show the proposed system's capability and effectiveness by using these testing data.

Image Type	Bands	Acquisition Dates	Pixel Size (m)
MSS	1,2,3,4	June 1979, May 1974, August 1981, July 1975	79, 158, 316, 632,1264,10k, 12k, 19k, 25k
TM	1,2,3,4,5,6,7	August 1985	30, 60,90,480,1920, 10k, 12k, 19k
SPOT	Panchromatic	September 1990	10, 20, 40, 160, 1800,2400, 10k, 19k, 22k, 25k

Table 5-1 Information of the Tested Remotely Sensed Images

How to test the retrieval effectiveness of large databases, especially a complicated image database such as the proposed system, is an important issue in information systems. Salton and McGill [Salton, et al, 1983] suggested a method, popular in evaluating the effectiveness of information retrieval systems, using precision and recall statistics. The basic idea of this method is, for a given query, let T be the total number of relevant items available, R_r the number of relevant items retrieved, and T_r the total number of retrieved items. Then precision P is defined in Equation 5-1, and recall R in Equation 5-2.

$$P = \frac{R_r}{T_r} \qquad (5\text{-}1)$$

$$R = \frac{R_r}{T} \qquad (5\text{-}2)$$

The two parameters are interdependent and one cannot be improved without sacrificing the other. In practice, information that needs to be retrieved may vary from situation to situation. In some cases, the users may require high recall, that is, the retrieval of almost everything that is likely to be of interest, while in other cases, the users may prefer high precision, that is, the rejection of everything likely to be useless. A good system should be the one which is able to, in most cases with everything else being equal, exhibit both a high recall and high precision and has ability to adjust recall and precision based on the user's requirements.

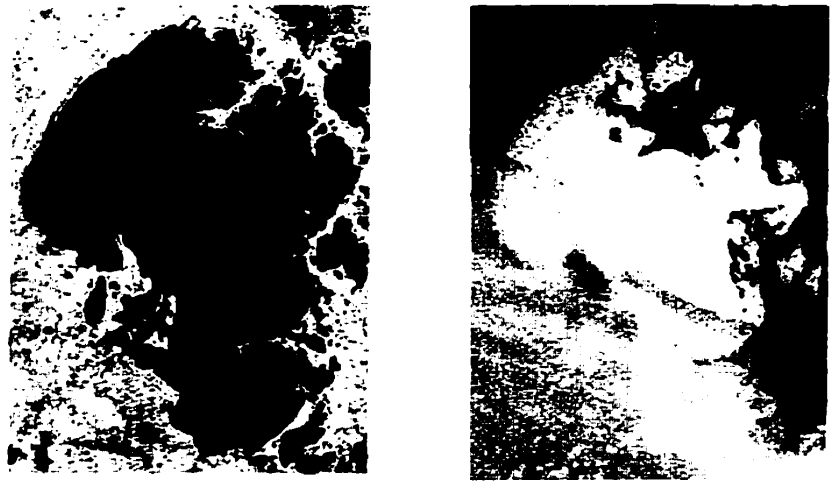
For the experiments, which items are relevant for each test case are decided beforehand. The precision P and recall R are computed for the following queries:

- Retrieval of images by QBS
- Retrieval of images by QBC

It should be noted that in these experiments, only query and retrieval performance of QBS and QBC are tested. Another important performance parameter, response time, won't be addressed. This is because the testing image database is too small to identify the response time differences in terms of query on spatial indexed objects and non-spatial indexed objects, clustered tables and non-clustered tables, partitioned tables and non-partitioned tables, etc. In future work, response time tests on real applications with large image databases which include large amounts of images should be investigated using timing functions available both in C language and RDBMS built-in function library.

5.4.1 Analysis of QBS Test Results

Figure 5-7 shows the example images used for QBS. The left image was acquired by satellite on June 1979. In the test database, there are 12 other images that were acquired in June or July in the same or nearby area. These 12 images are expected to be retrieved as the results of QBS based on the given image. The right image is the image that covers the same area as the left one, which was acquired on May 1974. The lakes in this image are covered by ice because it was still very cold in the spring. Table 5-2 shows the test results and Figure 5-8 is the corresponding graphs. In Table 5-2, T is decided beforehand. It's only used for the experiments. The users don't have to know this parameter beforehand when they invoke queries.



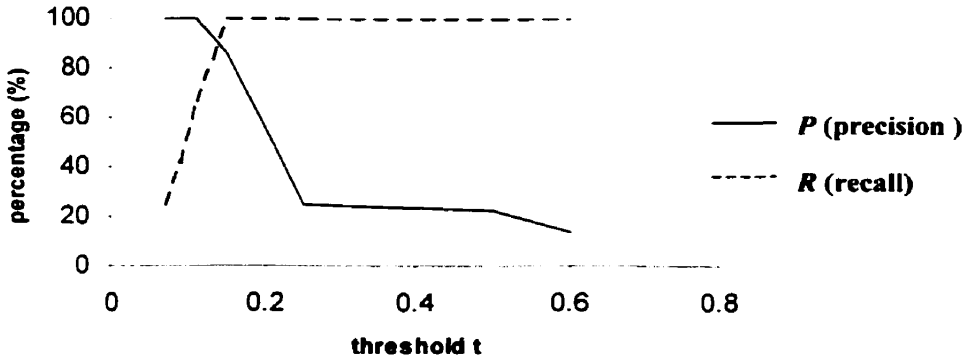
(a) **(b)**

(Lake Claire, dimensions: 655p x 896ℓ, pixel size: 79m
Note: p stands for pixel, ℓ stands for line)

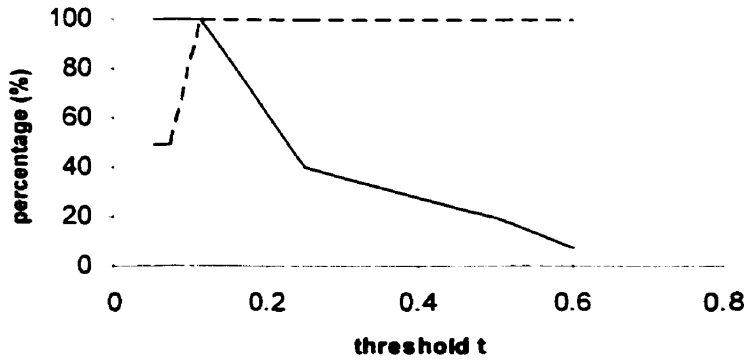
Figure 5-7 Example Images for QBS

example image	t (threshold)	T (decided beforehand)	R_r	T_r	P(precision,%)	R (recall, %)
Lake Claire MSS B421, June, 1979 Northern Alberta	0.07	12	3	3	100	25
	0.09	12	5	5	100	41.7
	0.11	12	8	8	100	66.7
	0.15	12	12	14	85.7	100
	0.25	12	12	49	24.5	100
	0.5	12	12	54	22.2	100
	0.6	12	12	87	13.8	100
Lake Claire MSS B421 May, 1974 Northern Alberta (ice on the lakes)	0.05	6	3	3	100	50
	0.07	6	3	3	100	50
	0.11	6	6	6	100	100
	0.25	6	6	15	40	100
	0.5	6	6	31	19.4	100
	0.6	6	6	79	7.6	100

Table 5-2 Results of Experiments with QBS



(a) Graph of the first QBS query



(b) Graph of the second QBS query

Figure 5-8 Test Result Graphs of QBS

From Table 5-2 and Figure 5-8, the following conclusions can be drawn:

1) $t<0.15$ in the first query and $t<0.11$ in the second query: high precision and low recall are obtained. That means when t is in this range, no "false alarms" (i.e., all retrieved images are qualified query selection criteria).
2) $t = 0.15$ in the first query and $t = 0.11$ in the second query: both precision and recall are high, which are the results we expected. The t at this point is called the *best threshold.*
3) $t>$ the best threshold: good recall is always yielded with sacrificing precision, that is, no "false dismissals" (i.e., all images qualifying query selection criteria are retrieved). Obviously, the larger the t, the more non-relevant images retrieved.

In conclusion, the effectiveness of QBS retrieval is significantly based on the selection of the similarity measure threshold t. The t can also be used as the tool to adjust precision and recall based on the user's specific requirements. It should be noted that the users' knowledge and understanding of the images they are querying and the images stored in the database can affect the selection of threshold. The more knowledge and understanding of images the users have, the quicker and more accurate the users find the best threshold. When working with the proposed system, the basic suggestions of the selection of threshold t are:

- In most cases, the purpose of the QBS is to find the best threshold, at which both precision P and recall R of the queries are high. The basic idea is to start with t=0.01 and adjust t steps by 0.01.
- The definition of satisfactory query results of QBC changes from situations to situations. Some time the users need high precision P while some time the high recall R is preferred.

Therefore, it's hard to define the rule of threshold t selection appropriate for each situation. The basic idea is as follows: if the result *P* is low, decrease the t and if the *R* is low, increase the *t* till the result *P* and *R* are acceptable.

Also notice that in the second query, precision and recall both hit 100% when t = 0.11. This is because the second example image was acquired in May. In that season in Northern Alberta in 1974, the lakes were still covered by ice. From Figure 5-6 (b), we can see the color of the whole image looks simpler compared to the image acquired in July. The image with snow and ice are easily differentiated from the images without snow and ice in similarity calculation because snow and ice usually have higher reflectance than other objects.

It is important to note that QBS is not an exact match query method as text-based query. It is an approximate query, sometimes called fuzzy query, which acts as a filter to reduce the set of images returned to the users. With such narrowed image set and with the visual query interface developed in this research, it is quite easy for the users to quickly scan the resulting images and quickly discard the non-relevant images. For large image database with hundreds and thousands of images, this technique is highly required and very useful in terms of significantly reducing search time and query response time.

5.4.2 Analysis of QBC Test Results

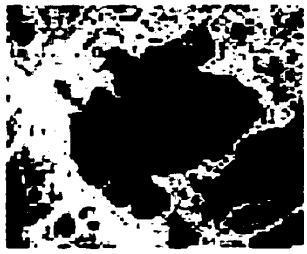
QBC is more complicated than QBS. As described before, the preprocessing techniques, such as the result of image classification, content extraction and calculation, directly affect the query results. In this experiment, it is assumed that the image preprocessing results are good

enough. Therefore, only the effectiveness of the query method and algorithm is tested.

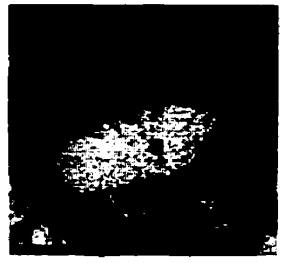
Figure 5-9 shows the examples of object used for QBC. The example object for the first two queries is Lake Claire, located in Northern Alberta, whose area is almost 1517km^2. Table 5-3 shows the test results and Figure 5-10 shows the corresponding graphs.

(a) Lake Claire: June 1979
dimension: 896p x 655l
pixel size: 79m

(b) Lake Claire: May 1974
dimension: 854p x 641l
pixel size: 79m



(c) Baril Lake: June 1979
dimension: 468p x 340l
pixel size: 79m



(d) Baril Lake: May 1974
dimension: 428p x 352l
pixel size: 79m

Figure 5-9 Examples of Objects for QBC

example object	t (threshold)	T (decided beforehand)	R_r	T_r	P (precision, %)	R (recall, %)
Lake Claire June, 1979 Area:1517±7%km^2	15	18	4	4	100.00	22.22
	16.5	18	16	16	100.00	88.89
	18	18	16	16	100.00	88.89
	21	18	16	16	100.00	88.89
	36	18	16	16	100.00	88.89
	40	18	18	32	56.25	100.00
	50	18	16	36	44.44	88.89
Lake Claire May, 1974 (ice on the lake)	18.2	18	12	12	100.00	66.67
	19	18	16	16	100.00	88.89
	25	18	16	16	100.00	88.89
	35	18	16	16	100.00	88.89
	45	18	18	39	46.15	100.00
	60	18	18	56	32.14	100.00
Baril Lake June, 1979 Area: 119±3%km^2	1.1	14	2	2	100.00	14.29
	1.5	14	2	2	100.00	14.29
	1.7	14	12	12	100.00	85.71
	2.4	14	12	12	100.00	85.71
	3	14	14	18	77.78	100.00
	3.5	14	14	27	51.85	100.00
	4	14	14	48	29.17	100.00
Baril Lake May, 1974 (ice on the lake)	1.31	14	2	2	100.00	14.29
	1.5	14	2	2	100.00	14.29
	1.89	14	12	12	100.00	85.71
	2	14	12	12	100.00	85.71
	2.5	14	12	12	100.00	85.71
	3	14	12	12	100.00	85.71
	3.35	14	14	18	77.78	100.00
	3.5	14	14	32	43.75	100.00
	4	14	14	55	25.45	100.00

** **Note: In the four queries, when t< the 1st number, 15, 18.2, 1.1, 1.31 respectively, R_r=T_r=0**

Table 5-3 Results of Experiments with QBC

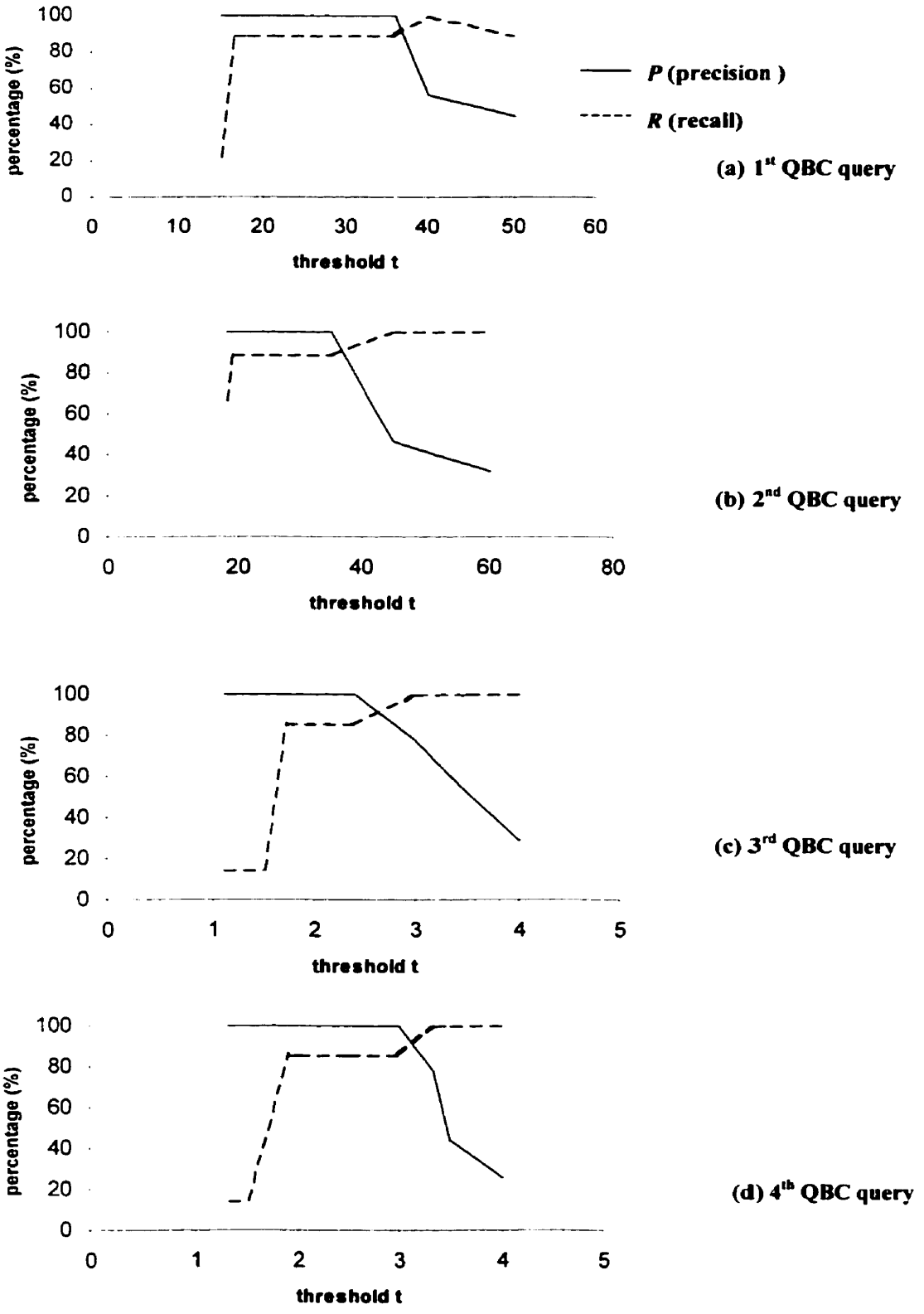


Figure 5-10 Test Result Graphs of QBC

From Table 5-3 and Figure 5-10, the following conclusions can be drawn:

1) t<15 in the first query and t<16.5 in the second query, no images are retrieved.
2) t = 15 in the first query, 4 images are retrieved and they are all relevant items, i.e., no false alarms.
3) 16.5<t<36 in the first query and 18.2<t<35 in the second query are the best threshold range in which both precisions and recalls are better than 85%.
4) No point at which both precision and recall can hit 100%. This is because, there two images, whose pixel size is almost 19km, which contain, for example, Lake Claire. In these images, Lake Claire is too small to be included in the polygon comparison. That is, the shape of the polygon has changed a lot which cause the big change of polygon parameters calculation. Figure 5-11 shows an example of Lake Claire whose pixel size is 25 km or so. This is the multiresolution problem as discussed in the last chapter. A conclusion can be drawn from this testing that, in the multiresolution, multiscale image database, there is a limitation for QBC to be appropriately applied. We can see when t>45 in the first query, the recall is 100% which means even the two images with 19km pixel size are retrieved with sacrificing precision.

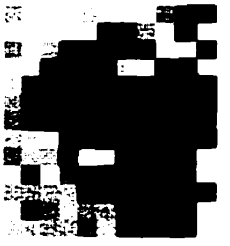


Figure 5-11 Lake Claire in 25km Pixel Size (dimension: 13p x 11*l*)

The example object of the second query is Lake Claire in May, covered by ice that results in shape change (Figure 5-11). The shape change might cause problems in QBC that would result in big differences of content calculation between the lake in this image and the ones in other images. From Table 5-3 and Figure 5-10, it is glad to find that the test results are almost the same as that of the first query. That means the QBS performs very well to retrieve the similar objects even from images taken in different seasons. That means that this method performs well in handling multitemporal images.

The example object of the third and fourth queries is Blair Lake, a small lake compared to Lake Claire. The test resulting data in Table 5-2 and the Figure 5-9 show that the results are almost as same as the first two queries.

In conclusion, QBC performed very well. Also, the effectiveness of QBC retrieval is significantly based on the selection of the threshold t. From the test results, another conclusion can be drawn that the smaller the example object, the more sensitive the results affected by the change of threshold t. Based on results of extensive testing (some test results are not shown in the Table 5-3) and QBC algorithm, the suggestions for threshold t selection and images are given as follows:

1) Same as QBS, in most cases, the purpose of the QBC is to find the best threshold, at which both precision P and recall R of the queries are high. Table 5-4 shows the basic rules for the selection of threshold t for QBC when working with the proposed system.

area of example object (km^2)	t starts with	adjust pace of t
500	1	0.2
500<area< 1000	5	1
area> 1000	10	5

Table 5-4 Rules of Selection of the Threshold t for QBC

2) If the result P is low, decrease the t and if the R is low, increase the t till the result P and R are acceptable.

3) In multiscale image database, there should be a limitation to the object size. If an object in an image is too small to be identified by classification method, such as the Lake Claire in Figure 5-11, it is suggested to leave it out of the QBC. Therefore, in the Image_Content table of large image database, there is a field including information whether this polygon of the object is appropriate to be involved in the QBC queries. It is a Yes/No field and it cannot be null. Yes means it can be involved while No means it is to be left out. Every time a new edge extraction result is stored in the database, this field must be populated.

Similar to QBS, QBC is not an exact match query method. It's still used the most as a filter to narrow down the image set returned to the users.

From the above discussion, we can see that, compared to traditional query methods, such as text-based query, color-based query, QBS and QBC are complicated query methods, Because of the complexity of these methods, they may not be suitable for all applications. The anticipated users of these methods are supposed to have basic remote sensing and image processing knowledge. The users without any remote sensing knowledge may not get satisfactory query results when they use these complicated query methods. For example, if a

user doesn't know the characteristic differences of radar imagery and TM imagery, he/she may compare them using QBS and the query results may not be meaningful. Therefore, the objective of these two methods is to help the experts in areas of environmental assessment and monitoring, global change detection and monitoring, agriculture, etc., who work with large image databases which have huge amounts of images, to reduce the sets of selected images for further analysis.

5.5 Advantages and Disadvantages of the Prototype

From the results discussed in previous chapters and the analysis of test results presented above, it can be seen that the prototype has the following advantages and disadvantages:

- ***Advantages:***

1) It provides a user-friendly, powerful, easy-to-use Graphic User Interface that can perform browsing, query navigation, query refinement, data editing and management without knowing any SQL syntax. The GUI can also support visual query such as Query by Similarity and Query by Content other than text-based query and retrieval.
2) It fully integrates advanced compressing, indexing, clustering and partitioning techniques. These advanced techniques improve the system performance significantly in terms of, especially, database management, search speed, response time of query and retrieval, etc. This advantage will become more critical as a database grows larger.
3) It supports similarity and content based query capabilities for large images in addition to supporting traditional queries and retrieval.

4) The extensibility of the system is comprehensive and straightforward because of the advanced techniques described above in 2) and also because the function library can be easily extended by adding application specific and user defined functions to improve the system performance.

- ***Disadvantages:***

1) The robustness, functionality and system performance are not optimized at this stage due to time limitations.
2) More image processing methods such as radiometric and geometric correction, appropriate filtering, edge detection methods for infrared and radar images have to be implemented.
3) It's not easy for inexperienced users, who have no knowledge of large images especially remotely sensed images, to get satisfactory query results in a reasonable time period. The knowledge management of large images should be integrated into the proposed system to help the users perform queries more efficiently and effectively. Chapter 6 will discuss this issue in details.

These limitations of the prototype can be minimized through various enhancements that are left as the future work of this research.

5.6 Summary

In this chapter, a prototype large image database development was discussed. Its user-friendly, powerful, easy to use GUI allows the users to browse the database, query database

and manage data without knowing SQL syntax. With a visual query interface, it's quite easy for users to perform Query by Similarity and Query by Content and query refinement. The function library provides image processing, query preprocessing and some standard database operation functionality and it is extensible to allow the users to add their specific functions to improve the system performance. Initial test results show that the prototype system perform QBS and QBC well. The selection of a threshold in both query methods is very important and critical to effectiveness and efficiency of the large image retrieval. The advantages and disadvantages of the prototype system are also discussed in this chapter.

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

This research presented a methodology for enhancing the existing commercial relational database to support large images, such as remotely sensed images including satellite images and airborne images, scanned images and other digital images, and their related metadata in addition to traditional types in relational databases for GIS applications. Although many DBMS vendors have integrated image data type support in their products, none of them has abilities to effectively and efficiently manage large images, especially remotely sensed images due to:

1) lack of effective data management utilities,
2) lack of effective indexing methods,
3) lack of image processing abilities,
4) lack of powerful visual query methods that are required for large image queries and retrievals.

Though, in theory, object-oriented database management systems (OODBMSs) are more appropriate than RDBMS in terms of large image support due to their advanced data abstraction technique, object identity supporting, class hierarchy, multiple inheritance and data encapsulation, OODBMS was not chosen as the based DBMS of this research mainly because of two reasons. One is that OODBMS is evolving but is far from mature to be employed in the industry. The other is that RDBMS is still the most popular DBMS model

used by industry and will be in the next several years, or even in the next decade. Therefore, the problems of large image support have to be overcome in RDBMSs.

The principal objective of this research was therefore to design and develop a Large Image Database that can extend the traditional relational databases with the following capabilities:

1) effectively and efficiently manage large images and their related metadata in a single database,
2) improve system performance by using advanced indexing, partitioning and clustering techniques,
3) allow users to easily add new functions and their knowledge into the function library to improve database performance,
4) provide a user-friendly graphic user interface that allows users to perform database browsing, query navigation, similarity and content based queries and retrievals and manage databases and data without knowing SQL syntax.

To this end, Oracle8™ has been applied as the base DBMS in this research mainly because, firstly, it's the most popular and most powerful DBMS in the market and, secondly, it provides advanced techniques, such as partitioning, rich query processing techniques, sophisticated SQL optimizer and extended backup/recovery techniques, etc. to support very large database management.

There are a number of significant benefits to the methodology presented here:

1) By storing large images, their related metadata and other data types in one database, users benefit from the perception that they have access to a seamless information environment,

no need of considering differences in data sources, information types, storage devices and processing techniques. In addition, data integrity, transaction, security and recovery can be managed using one DBMS.

2) With a visual environment, users can easily perform special image query, such as Query by Similarity and Query by Content other than traditional text-based queries and even perform more complicated queries based on the resulting image set of QBS and QBC.

A number of conclusions can be drawn from the work presented herein:

1) Large images and their metadata are successfully integrated into one database. Metadata management of remotely sensed images in the proposed system meets the FGDC metadata standards. This is very important and is mandatory for future work in areas like data distribution, data sharing and Web-based large image database implementation.
2) A user-friendly graphic user interface has been designed and developed to provide capabilities of database browsing, query formation and execution, progressive query result refinement, database and data management without knowledge of SQL syntax. These tools have made the proposed system easy to use and comprehensive.
3) Some image processing, query preprocessing and standard database operation functions have been developed in the function library. Users can easily add application specific functionality to improve the system performance.
4) Query by Similarity and Query by Content have been implemented for users to perform query based on image information similarity and objects contained in the images. The test results have showed that using these query methods can yield effective and efficient retrievals.

5) The proposed query methods, QBS and QBC, have limitations in terms of handling boundary problems, infrared and radar images, multiscale and multiresolution images. More future work needs to be done to overcome these problems.

6.2 Recommendations

1) The proposed system should be extended to support other large objects and multimedia objects, such as digital terrain models (DTMs), time series object, audio and video clips, etc. Supporting these data types requires the system to have real-time modeling and query capability.

2) Rules are key components of information systems. A rule is a predicate and action pair in the form of <predicate, action> [Zhou, 1995]. When the predicate is true, the action will be automatically executed. For example, the rule "two satellite images from different sensors (e.g. one from SAR and the other from TM) cannot be used for comparison using QBS" can be enforced when the users without remote sensing knowledge invoke the QBS query to prevent wrong query results. With rule support, large image databases will be able to automatically retrieve the most suitable data set based on the user's current application context and/or scale. More untrained users could use these advanced and complicated QBS and QBC methods. Furthermore, the data and database integrity will be greatly extended.

3) Different image format support is critical to an image database. So far, the proposed system supports the image formats that can be handled by Oracle8^{TM}. Hierarchical Data Format (HDF) is good at handling the exchange of different graphic file formats [Provins,

1998]. The proposed system should be enhanced to include HDF operations and functions.

4) Uncertainty handling should be enhanced in the proposed system since uncertainty information aids the users in determining the fitness of the data to their particular applications. Furthermore, it helps to determine the threshold value t in QBS and QBC calculations.

5) Tests on real applications with large image databases which include large volumes of images, more complicated processing requirements should be carried out to evaluate the actual performance of the proposed system. The evaluation may lead to some modifications of the system to optimize and improve the system performance.

6) Finally the integration of image processing systems, GISs and RDBMSs as a whole part will be one future direction of this research.

REFERENCES

Alesheikh, Ali Asghar [1998], Modeling & Managing Uncertainty in Object-Based Geospatial Information Systems, Doctoral Dissertation, Department of Geomatics Engineering, UCGE Report No. 20121, The University of Calgary.

Anderson, Jean T. and M. Stonebraker [1994], Sequoia 2000 Metadata Schema for Satellite Images, SIGMOD Record, Vol. 23, No. 4, December 1994, pp, 21-27.

ANSI X3.135-1989 [1989], Database Language SQL with Integrity Enhancement. American National Standard Institution.

ANSI X3. 135-199X [1993], Database Language SQL. ISO and ANSI SQL3 Working Draft.

Biliris, A. [1992], The Performance of Three Database Storage Structures for Managing Large Objects. Proceedings of the 1992 ACM SIGMOD. International Conference on Management of Data. San Diego, CA, pp. 276-285.

Blais, J.A.R, D. Gourdeau and C. Stewart [1997], The Crown of the Continent Ecosystem Data Atlas Project – a Geomatics Perspective. Geomatica, Vol. 51, No.3, pp. 235-245.

Brathwaite, Kenmore S. [1991], Relational Theory -- Concepts and Applications, Academic Press, Inc.

Chang, S.K. and C.C. Yang [1982], Note: Picture Encoding Techniques for a Pictorial Database. Spinger-Verlag, pp.33-53.

Date, C.J. [1995], An Introduction to Database Systems, Sixth Edition, Addison-Wesley.

Federal Geographic Data Committee [1997], Geospatial Metadata, URL: http://www.fgdc.gov/publications/documents/metadata, March 1997.

Fu, Ada [no date], Content Based Image Indexing, manuscript, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

Gonzalez, Rafael C. and Richard E. Woods [1992]. Digital Image Processing, Addison-Wesley Publishing Company.

Guttman, Antonin [1984], R-trees: A Dynamic Index Structure for Spatial Searching, Proceedings of ACM SIGMOD, pp, 47-57, June 1984.

Jahne, Bernd [1991], Digital Image Processing -- Concepts, Algorithms and Scientific Applications, Springer-Verlag.

Larouche, Christian [1995], Automation of Photogrammetric Operations Using Advanced Digital Image Matching Techniques, Doctoral Dissertation, Department of Geomatics Engineering, UCGE Report No. 20085, The University of Calgary.

Levine, John [1994], Programming for Graphics Files in C and C++, John Wiley & Sons, Inc.

Lillesand, Thomas M., and Ralph W. Kiefer [1987], Remote Sensing and Image Interpretation, John Wiley & Sons, Inc., New York.

Oracle [1996], Oracle™ Unleashed, SAMS Publishing.

Oracle [1998], Oracle8i™ Concepts, Oracle Corporation.

Petrakis, E. and S. Orphanoudakis [1993], Methodology for the Representation, Indexing and Retrieval of Images by Content, Image and Vision Computing, Vol. 11, No. 8, pp.504-521.

Petrakis, Euripides G.M. and Christos Faloutsos [1994], Similarity Searching in Large Image Databases, University of Maryland Institute for Advanced Computer Studies. Dept. of Computer Science, Univ. of Maryland, December 1994.

Provins, Dean [1998]. Hierarchical Data Format: A Platform for Research Satellite Geodesy and Application, Department of Geomatics Engineering, Winter 1998.

Rennhackkamp, Martin [1997], Extending Relational DBMSs, DBMS. URL: http://www.dbmsmag.com/9712d15.html

Robinson, V.B. and D.S. Mackay [1993], On Heterogeneous Geographic Information Systems, Architectures, Spatial Data Models, Transactions and Database Languages. In Towards SQL Database Languages for Geographic Information Systems, Edited by: Robinson, V.B. and H. Tom, pp.1-35.

Salton, G and M.J. McGill [1983], Introduction to Modern Information Retrieval. McGraw-Hill.

Schowengerdt, Robert [1997], Remote Sensing Models and Methods for Image Processing, Academic Press.

Stonebraker, M. and M. Olson [1993], Large Capacity Object Servers to Support Global Change research. Sequoia 2000 Technical Report 1. University of California at Berkeley.

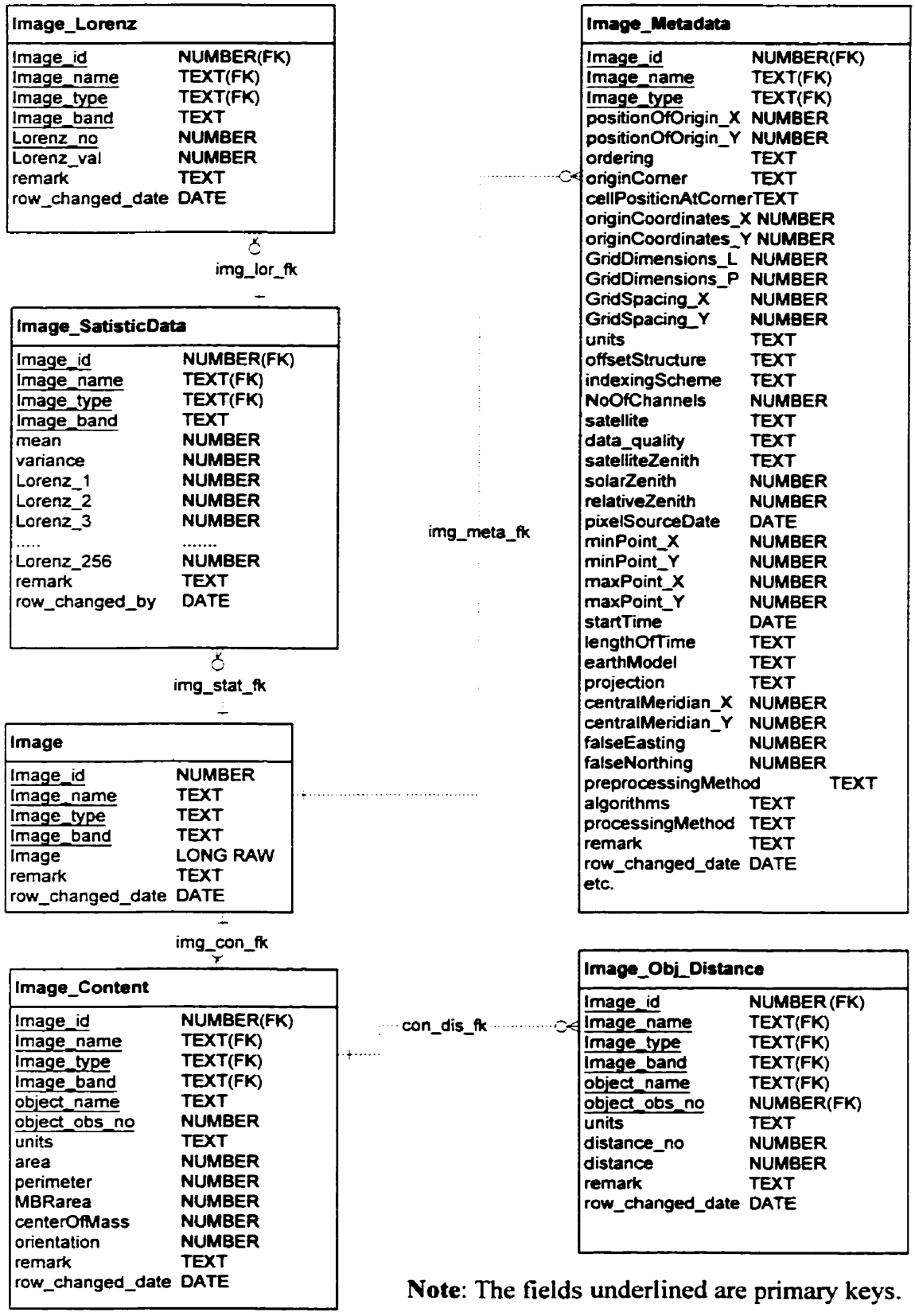
Surveys and Resource Mapping Branch [1994], Spatial Archive and Interchange Format: Formal Definition Release 3.1, Province of British Columbia, Ministry of Environment, Lands and Parks, April 1994.

U.S. Geological Survey [1991], The 1991 Conterminous U.S. AVHRR Biweekly Composites, README file from the CD-ROM, EROS Data Center, U.S. Geological Survey, Sioux Falls, SD.

Zhen, Chao [1996], An Object-Oriented Prototype for an Environmental GIS of the Creosote Site on the Bow River, Calgary, Master's Thesis, Department of Geomatics Engineering, UCGE Report No. 20106, The University of Calgary.

Zhou, W., W.Brunner and S. Wilkinson [1991], Full Integration of Remote Sensing Functions into a Vector Based GIS. Proceedings of GIS/LIS' 91, Oct. 28-Nov.1, Atlanta, Georgia, pp, 795-804.

Zhou, Wenjin [1995], Large Object Support Using an Object Oriented Approach in Spatial Information Systems, Doctoral Dissertation, Department of Geomatics Engineering, UCGE Report No. 20084, The University of Calgary.



Note: The fields underlined are primary keys.