Monitoring and Detection of Hotspots using Satellite Images

By

Juliana Binti Mohamad Yusof

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Business Information System (Hons)

JUNE 2006

Universiti Teknologi PETRONAS
Bandar Sri Iskandar
31750 Tronoh
Perak Darul Ridzuan.

CERTIFICATION OF APPROVAL

Monitoring and Detection of Hotspots using Satellite Images

By

Juliana binti Mohamad Yusof

A project dissertation submitted to the
Business Information System Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(BUSINESS INFORMATION SYSTEM)

Approved by,

(Mr. Justin Dinesh Devaraj)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
JUNE 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

JULIANA BINTI MOHAMAD YUSOF

ABSTRACT

Nowadays, the usage of optical remote sensing NOAA-AVHRR satellite data is getting familiar as it is known can save cost in order to capture a wide coverage of ground image. The captured images are meaningful after several processes done over it to produce hotspot detection. Developing a specific database to store information of Hotspots (LAC images) would make data management and archiving purpose in more efficient and systematic way. Real-time data gathered are monitored countries such as Malaysia, Thailand, Singapore, Indonesia and Brunei within the region of NOAA Satellite coverage area. PostGIS, PostgreSQL, Mapserver and Autodesk MapGuide Studio software are to be studied as a guide to develop a system with simple database using object-relational database management system to store raster and vector images. This paper describes a solution for efficient handling of large raster image data sets in a standard object-relational database management system. By means of adequate indexing, retrieval techniques and multi resolution cell indexing (Quad-Tree) can be achieved using a standard DBMS, even for very large satellite images. Single image will be divided equally into 64 small squares (3 levels of image hierarchy - each level has 4 sub images of the higher image). Partial information of Daily Haze report (processed Hotspot on image map) produces by NREB can be viewed using web-based application. The final product of this project is a web-based application for displaying Hotspots on maps (combination of raster and vector images) with the ability to search record from database and functions to zoom in or zoom out the map. The objective of this paper is also to show the way satellite images and descriptive information are combined and amalgamated to form an Internet or Intranet application.

ACKNOWLEDGEMENT

Heartfelt thanks to the ALLAH Lord Almighty for given me strength, wisdom and patience to finally complete my Final Year project. I would like to take this opportunity to convey my greatest appreciation to all people who have been involved directly and indirectly during the accomplishment of this Final Year project.

First of all, I would like to thank Universiti Teknologi Petronas FYP Committee members who gave me the opportunity for me to do research on this project by approving my Final Year Project title proposal and for managing this course with full-dedication and proficiently.

Thank you so much to my supervisor, my most gracious mentor and guide Mr Justin Dinesh Devaraj Daniel for supervising the on-goings of my project and guided me in through out the accomplishment of my Final Year project. I have tremendously enjoyed with lots of new experiences in GIS and will treasure them all my life.

Not forgotten to Mr Rusdan Abas, AECO in NREB Kuching, for his help in providing data for my project and for his willingness, commitment and his up-most collaboration and assistance in helping and guide me through out understanding the processes of monitoring hotspots in Malaysia. It is with his patience and understanding made me able to come out with the topic's proposal hence completing the user requirements for this project the overview of system. Thank you for the sharing of such expensive information, knowledge and experience regarding my project matter.

Last but not least, a big thanks to my parent for being very supportive and understanding all along. I feel a deep sense of gratitude for them who had formed part of my vision and taught me the good things that really matter in my life. My family also provides a persistent inspiration for my journey in this life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE
INTRODUCTION

1.1 Background

Over the last 10 years, GIS (Geographical Information Systems) has become an essential part of a wide range of computerized applications in a variety of fields, including environmental protection, urban and regional planning, transportation, demographic analysis, military operations, and optimal location planning.

Remote sensing to detect fires has been used for the past 15 years and was critical in 1997 to locate fires, track smoke plumes and to identify large plantations as major fire. Here, we assess the feasibility of using remote sensing to estimate surface-level concentrations of particulate air pollution generated from vegetation fires. This project is an initial test of using remote sensing as a tool to identify such large-scale vegetation fire.

Following the serious haze Sarawak experienced in 1997, the Sarawak State Government responded quickly by setting up forest fire detection system. This has been one of the main roles of the CREM, which is situated at the NREB office at Petra Jaya, Kuching. The procedure is carried out using satellite reception system that obtains data from NOAA Polar Orbiting Satellites series (NOAA-12 and NOAA-14) and Fung Yuen Satellite (FY1C and FY1D). These satellites are located approximately 810 to 860 km above the Earth's surface and take 102 minutes to complete one orbit.

The usage of remote sensing is getting wider as it can offer a better solution which is far cheaper than other methods in monitoring and detecting fire in a wide coverage. This information can be used by State Government as well as Federal Government and other organization (CREM in Kuching) as guidance for location of future monitoring stations within the region and to predict potential haze spots.

1.2 Problem Statement

i. CREM does not provide a proper archiving for the processed data.

ii. There is no backup and recovery tool provided for data stored. Data are stored in remote computers' hard disks and are only available across the office network.

iii. Hard to search processed images that are named according to its original file name given by Dundee Satellite system (less meaningful name).

iv. Hotspot daily report which consists of processed hotspot images map together with few other necessary information such as API (Air pollution Index), FWI (Fire Whether Index) and Wind Direction data is updated manually.

v. NREB website does not have the functionality to zoom in or zoom out the hotspot images.

1.3 Objectives and Scope of Study

The objectives are:

i. To use processed satellite images data, in combination with map layers provided, to develop LAC (Local Area Coverage) images.

ii. To develop a database to store information of Hotspots (LAC images) in more efficient and systematic way.

iii. To gather real-time data of monitored countries within the region of NOAA Satellite coverage area (Malaysia, Thailand, Singapore, Indonesia, Brunei).

The scope of studies are:

i. Analysis on real-time images used by CREM which are captured using remote sensing technology (NOAA-AVHRR and Fung Yuen).

ii. Study of DBMS in order to develop a simple database for storing raster images.

iii. Study on present systems available in CREM and the database to store raster and vector images using object-relational database management system.

iv. Display partial information of Daily Haze report using web-based application by thorough study on Mapserver and MapGuide. Developing a web-based application for viewing Hotspots on maps and to make it available online (intranet and later internet)

CHAPTER TWO

LITERATURE REVIEW AND THEORY

2.1 Satellite technology

Understanding of remote sensing images is one of the challenging problems in computer vision. Due to its complexity, knowledge-based systems have been found to be mandatory since the mid seventies e.g. Agin 1979 [1], McKeown et al. 1985 [2], Nicolin & Gabler 1987 [3], Matsuyama & Hwang 1990 [4] and Quint & Sties 1995 [5]. Different examples of knowledge based systems applied to the task of remote sensing image understanding can be found in the literature. AIDA Liedtke et al. 1997 [6]: a system for the knowledge based interpretation of remote sensing data, uses semantic nets for the explicit representation of the prior knowledge about objects expected in the scene. It exploits the knowledge base to generate a scene description assigning symbolic meanings to the image primitives. The information of a GIS database is used as partial interpretation to produce reliable hypotheses for the expected objects. This initial scene description is verified consecutively in the remote sensing imagery. Multiple sensors can be investigated simultaneously. The explicit knowledge representation eases the adaptation to different tasks. Koch et al.1997 [7] showed that regarding remote sensing data and maps as different kinds of sensors allows a similar approach for both in the domain of landscape interpretation. The prior knowledge about the landscape objects is represented explicitly by semantic nets. Important is that the scene analysis employs a partial interpretation derived from a Digital Landscape Model (ATKIS DLM 25/1). This partial interpretation is used to generate an initial scene description. Consecutively the scene description is verified in aerial images and maps. Interpretation proceeds iteratively mixing top-down and bottom-up strategies. Koelma & Smeulders 1999 [8] presented a blackboard infrastructure for the development of object-based image interpretation applications. The infrastructure is based on an abstract definition of important concepts in image interpretation: data objects, relationships, algorithms, strategies, and models.

High resolution satellites are promising technologies to bolster global transparency by providing unprecedented access to accurate and timely information on many important, perhaps controversial, developments. Due to the fact that a significant number in this satellite fleet are owned by private organizations, these can collect images of human activities and the environment and make them commercially available around the globe. There are now more than 15 commercial satellites with resolutions from 1 - 30 meters. For example, satellites such as IKONOS (launched in September 1999), ORBVIEW (OrbView–3 launched in Q2 of 2002), EROS and QUICKBIRD are privately owned and provide images with resolution of 1 meter or smaller. One-meter imagery enables the viewing of houses, automobiles and aircrafts, and will make it possible to create highly precise digital maps and three-dimensional fly-through scenes.

As low cost high resolution images become available, it is not unrealistic in near future to have information products such as that of Microsoft TerraServer in which a data warehouse of image pyramids constructed from aerial, satellite, and topographic images of the earth. With Terraserver, users can search the data warehouse by coordinates and place names, and can view the images with different resolutions by simply clicking on them. With such data warehouses built with images taken at closer time intervals, one can attain near real-time surveillance of a desired region.

Recent advances in remote sensing technology are making it possible to capture geospatial orthoimagery (i.e., imagery that created so that it has the geometric properties of a map) with a resolution of 0.3 meter or better. These images are available online and have been utilized to enhance real estate listings, military targeting applications, and other GIS applications. One of the key issues with these applications is to automatically identify man-made spatial features, such as buildings, roads and road intersections in raster imagery. Computer vision researchers have long been trying to identify features in the imagery [9]. While the computer vision research has produced algorithms to identify the features in the imagery, the accuracy and run time of those algorithms are not suited for these real-time applications.

Raster images can represent continuous values such as Elevation. Raster can also represent discrete values such as buildings, parking lots, and roads, or soil types. Categorical raster can also store additional attribute data.

Types of raster data:

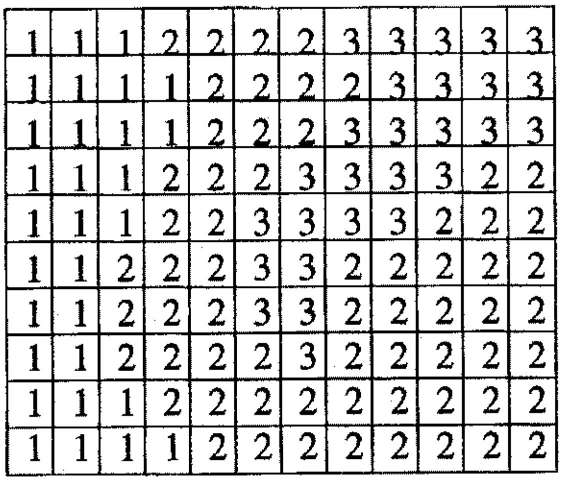
i. satellite imagery
ii. scanned aerial photographs
iii. digital orthophotos
iv. grids - ArcGIS raster files

Satellite imagery comes in raster format. Raster format is differing from vector as raster model stores cells in continuous space while the vector model represents features as x, y locations. These locations are connected together to make arcs (lines) and polygons (areas). As the content of the storage differs, a raster image uses much smaller amount of file size compared to vector images. Such satellite images are produced massively thus the conversion of raster-to-vector would not be the best alternative to store converted images (in term of storage space) although images in vector representation has much better resolution than raster. Therefore, main objective of this project is to develop a web-application to display hotspots map using both raster and vector images to provide a good quality presentation.

2.2 GIS Graphic Data Structures (Spatial data: Raster and Vector Images)

2.2.1 Raster

Raster data is organized as a matrix or grid; each row/column intersection is a cell or pixel based, in simple terms this is a data set of rows and columns of numbers where the position (column, row; x, y) is the location and the number (or character) is a code representing some attribute of the data. Each cell has a value, for example, an elevation. The size of each cell is generally determined by some type of header record for the file that describes the coordinate of the origin (row 1, column 1) of the file and the x,y dimensions of the cells in the file. They are a specific number of pixels high and wide, with each pixel representing a certain size on the ground; for example, Landsat satellite images are 185 x 185 km in size. Each pixel is 30 x 30 m in size.



1	1	1	2	2	2	2	3	3	3	3	3
1	1	1	1	2	2	2	2	3	3	3	3
1	1	1	1	2	2	2	3	3	3	3	3
1	1	1	2	2	2	3	3	3	3	2	2
1	1	1	2	2	3	3	3	3	2	2	2
1	1	2	2	2	3	3	2	2	2	2	2
1	1	2	2	2	3	3	2	2	2	2	2
1	1	2	2	2	2	3	2	2	2	2	2
1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	1	2	2	2	2	2	2	2	2

Figure2.1: A simple raster data set with three attributes

Raster data forces all features to be represented in grid cells with specific dimensions. Some areas are fairly well represented by raster data structure, provided the raster structure aligns with the feature orientation and the size of the features. We can increase the accuracy of spatial representation of features by increasing the resolution of the raster data (make the cells smaller). This is done however, at expense of increased storage space and processing costs on the computer.

Two of the most common ways raster data (pixels) are stored are as **run-length encoded** (a form of compression) and **quadtree** (hierarchical).

Run-length encoded

Lets say the first row of a raster data set looked like this: 22222333333445522223 3
One form of run-length encoding would store the data as data pairs with the first number being the repetition factor and the second the data value. The above line would code as:
526324254223. Some simple image compression algorithms operate like this.

Quadtree data structures

This data structure is used to represent data in layers that are build successively from pixels of the previous layer. A pixel in each higher layer is derived from the average or majority of 4 pixels from the lower layer. This is not a data reduction technique but provides a way to store data for rapid retrieval on display devices (the smallest layer for the display size is used for the display).

Raster comprises of simple data structure for storage (rows and columns) make it easy to analyze and compare to other rasters (differencing operations). Most form of imagery maps used on the Internet nowadays is raster images because modeling applications are much easier to program and implement using raster code.

One problem with using raster maps is that it is cumbersome to associate hyperlinks with objects on a raster map, in particular at a fine level (e.g. for each of a large set of buildings). Its irregular features are not well represented as to their true size and shape. Linear features are represented by rasters being turned on and take on a jagged appearance. This is both annoying and inaccurate. Another problem in raster data is deciding how to code each cell. Raster can be coded by majority rule or cell center.

Moreover, high quality images usually come with a very large storage requirements size even for simple data (note that complex landscapes can actually take more storage with vector data than raster).These maps are slow to download, in particular over low bandwidth links such as over a phone line and modem.

This performance problem is amplified by the fact that adding/removing a layer or zooming/panning on a raster map can only be done by requesting a new image from the server. Other issues with raster images are zooming to a low depth will pixilated the image; generally higher processing required to analyze the image to perform spatial queries and limited spatial accuracy depending on the quality and nature of the image.

2.2.2 Vector

Vector data is objects represented in vector data structures are determined by an x, y location in coordinate space. Vector data sets are composed of single points, lines, polylines or arcs (connected string of points), and polygons (series of coordinates that define an enclosed region). Vector also involves arc, series of line segments bounded by nodes at end points and vertices and topology, the way a vector GIS uses points, lines, and polygons to represent map features.

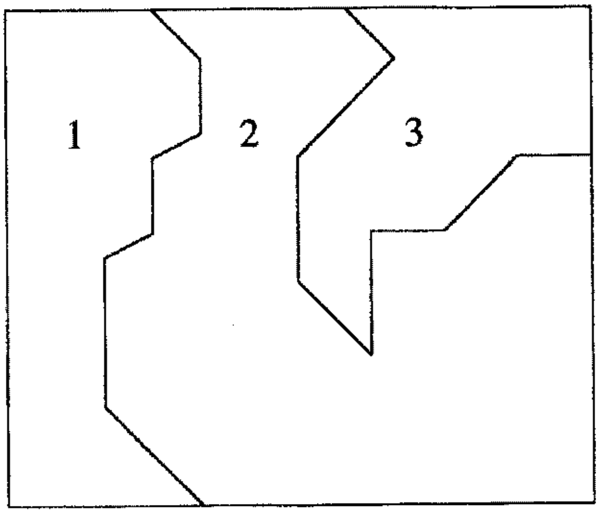


Figure 2.2: A vector layer with 3 polygons

The most common vector data structure utilizes arc/node topology. **Topology** can be defined as the organization of spatial relationships between features in a GIS. In layman's terms, topology is the way a GIS "knows":

1) where a feature is in relation to other features,

2) what parts of different features are shared (points, lines, nodes), and

3) how features share connectivity (gives us ability to move between features in network applications).

Vector data, more what we expect as map data such as points (e.g. cities and post offices), lines (e.g. roads and rivers) and polygons (e.g. city boundaries, suburbs and land parcels), are more suitable for this type of Internet application, especially when the application runs as a Java applet on the client side.

The most prominent advantage of using vector data is that it becomes possible to perform some spatial processing on the client side, such as zooming in and out, and turning on and off layers. With vector data, the map image does not pixilated when zooming in. It has better resolutions than raster generally on detailed landscapes and better spatial accuracy.

Grouping objects into layers and associating hyperlinks with spatial objects is a difficult task for raster maps but becomes trivial for vector data using the advantage of topology (can represent connectivity and interrelationships. In addition, vector data can be small and compress extremely well. New data is downloaded from the server only when it is not available in the client's cache (for example, when a new layer is required).

However, apart from those advantages, vector also has its disadvantages compared with using raster data that need to be dealt with carefully, namely, the overhead for rendering data on the client side where it is difficult to manage on a computer; slow to process complex data sets on low-end computers and potentially a very large amount of data needed for a map covering a large area.

Vector maps can be digitized and edited on tablets or on the screen, with automatic creation of topological information. *Digital terrain models* can be created by digitizing irregularly-spaced points or by sampling contour lines, with support for both regular and triangular grids. *Remote sensing image* geocoding can be made by means of ground control point location, and images can be registered with maps or with images.

2.2.3 Raster / Vector Conversion

Modern GIS software provides for conversion between raster and vector data structures. The conversion is needed to ease of modeling digitize data in vector format then convert to raster; converting vector to raster for **printing, plotting** and converting raster to vector to utilize **topologic data structures** (stream modeling). *Raster-to-vector* and *vector-to-raster* conversions also enable the mapping between the available formats with other data types in system. SPRING also *imports* and *exports* data from a number of formats, including: ARC/INFO, DXF, SPANS, TIFF, ERDAS, PCI, MaxiCAD and SGI/INPE.

The combination of the spatial and attribute data along with the creation, editing, data retrieval, and output capabilities is what makes up the sum total of a GIS.
Most of available GIS tools need to convert a raster image to vector first before it can display the images. However, my project is about developing an application to display raster images without bothering to make any conversions. Together with vector data, the presentation of Hotspots monitoring would be excellent.

2.3 GIS Database

A goal of GIS is to represent and store the graphic entities of mapped information along with relevant attributes in such a way to make all the data easily retrievable and manipulatable. This is done by taking advantages of the ways computers handle data in logical fashion through file and database structures. A brief overview of the ways computers can handle data is offered here.

The most basic file structure is a **simple list**. In this file structure, there really is no absolute ordering of the data. The data occur in the file in essentially the same way in which they were entered. Simple lists may start out in logical fashion but whenever modifications are made, they rapidly get out of order because new data are appended to the end of the list.

Ordered sequential files can be thought of as a rolodex in which we keep everything in alpha-numeric order. As new data are added, the file is restructured (sorted) to maintain that order.

Indexed file structures provide pointers to more efficiently search data. The most efficient system is to develop an index that is based on a commonly searched attribute in the database. A collection of files that are used for complex information organization is called a database. The software used for management and manipulation of databases is called a database management system (DBMS).

The recent advances in database technology have enabled the development of a new generation of spatial databases, where the DBMS is able to manage spatial and non-spatial data types together. Most spatial databases can deal with vector geometries (e.g., polygons, lines and points), but have limited facilities for handling image data. The software in GIS creates a database that keeps track of the relationships as lists of shared features. The database structure is designed to keep a list of all arcs and how they relate to the formation of each polygon. There are three types of DBMS's relevant to GIS: **hierarchical, network** and **relational**; the latter is most often the form of system adopted for complex GIS operations.

A GIS database is composed of all of the geographic/spatial information (maps, imagery) and associated attribute information (tables, reports) that are linked in such a way that we can extract either the spatial or attribute information by requests based on the location or characteristics of the data features either singly or as related to other features. There are two types of data in a GIS: **spatial** and **attribute** (tabular).

2.3.1 Attribute Data

These are usually data tables that contain information about the spatial components of the GIS themes. These can be numeric and/or character data such as timber type, timber volume, road size, well depth, etc. The attributes are related back to the spatial features by use of unique identifiers that are stored both with the attribute tables and the features in each spatial data layer. Attributes can be either *qualitative* (low med, high income) or *quantitative* (actual measurements).

The database allows us to manipulate information in many ways: from simple listing of attributes, sorting features by some attributes, grouping by attributes, or selecting and singling out groups by attributes.

The database software that we are using in lab is a relational database structure. This means that we cross reference feature attributes to their spatial definitions based on some common attribute stored in the data table for the attributes and graphics. We can select one or more graphic features by use of a query of some characteristic of interest in the feature attribute table of the database. And, since the reference from graphics to attributes works both ways, we can select one or more spatial graphic features on our computer screen and have the software give us the associated attributes. The relational qualities of the database go even further than these simple examples. We frequently attach other external tables to our original data sets and relate them to the existing data by common attributes.

Benefits of Relational Database Approach

1. Convenience - easy access
2. Reduce redundancy - cross referencing reduces repeated data entries
3. Shareable - cross organization sharing, multiple uses of data
4. Flexible analysis options - multiple analysis options, expansion of analysis capability
5. Standardization - controlled formats for data entry / manipulation reduce inconsistency and errors.

2.3.2 Spatial data (Raster format)

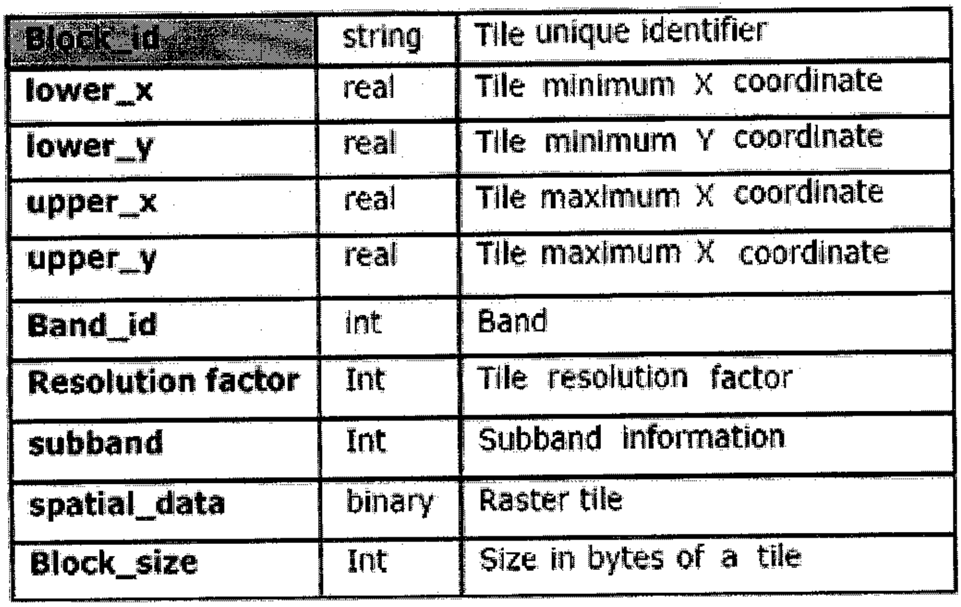
This discussion will focus on spatial data representation on computer systems, the pros and cons of different data structures, and some general comments on use of the data. First consider the following definitions:

Map is a paper representation of features that can be depicted based on their spatial relationships. Here we will be dealing with maps of the earth's land surface.

Coverage is the digital form of all or part of a map. In our work, a coverage is usually depicts one major *theme* (ArcView term) such as landuse, roads, streams. The more specific we make the coverage, the more flexibility we build into the analysis capabilities of the GIS.

Data Structure is the form (format) of the data as stored and manipulated on the computer. In order to get the data into the computer, it must be scanned or digitized, edited, and converted to the final form for the GIS (raster or vector).

In order to store the raster data in a DBMS, previous works in the literature (Patel, Yu et al. 1997; Reiner, Hahn et al. 2002) have shown than a combination of *multi-resolution pyramid* and a *tiling* scheme is the most appropriate strategy for handling large image files. The tiling scheme is used as a spatial index, such that when retrieving a section of an image, only the relevant tiles will be retrieved and decompressed. The multi-resolution pyramid is very useful for visualization of large data sets, to avoid unnecessary data access. This approach was adopted in TerraLib [12]. Each image is associated to a database table, called the "raster table", where each record stores a tile (or block) of the image. The fields of the raster table are shown in Table 1.



Block_id	string	Tile unique identifier
lower_x	real	Tile minimum X coordinate
lower_y	real	Tile minimum Y coordinate
upper_x	real	Tile maximum X coordinate
upper_y	real	Tile maximum X coordinate
Band_id	int	Band
Resolution factor	Int	Tile resolution factor
subband	Int	Subband information
spatial_data	binary	Raster tile
Block_size	Int	Size in bytes of a tile

TABLE 1 – The Raster Table in TERRALIB

For image storage in a TerraLib database, each band of the image is decomposed in a set of disjoint tiles with width *W* and height *H* given in number of pixels. Each tile is stored in the field *spatial data* (as a long binary) of a record in a raster table. Tiles can be compressed before storage. Global information about a raster layer, such as tile dimension, compression technique, number of rows and columns, number of bands, bounding box, and pixel resolution is stored in a complementary table. The bands of the image are divided in blocks of dimension 2n x 2m (the default is 512 x

512) to simplify the construction of a multi-resolution structure. The starting point for the tiling division must correspond to a known geographical reference.

This allows a definition of a function that returns the same block identification for all the pixels that belong to a given tile, and is used as primary key of the raster table. For example, consider a 100x100 image with a pixel resolution of 1 meter, where the lower-left pixel has a geographical coordinate of (20,20) and the upper right pixels of (119,119), in the cartographical projection associated to the image. The decomposition of this image (represented by the gray rectangle) in tiles of 64x64 pixels will generate the four tiles.

In order to derive an object-oriented model, it is necessary to distinguish between the *conceptual level* (where the abstract entities are defined) and the *implementation level* (where the graphical representations are constructed) [10].

The implementation of object-oriented modelling in developing any database is still unstable, it is not widely chosen by most database administrator. Thus, for this project, I would concentrate on developing a relational-object database management system (ORDBMS) where it combines the functions of object-oriented database modelling and relational database modelling. Nowadays, ORDBMS approach is easy to be put into practice and it is commonly use in developing databases.

Image processing systems, such as ERDAS Imagine, are widely used in remote sensing. They offer advanced image processing algorithms, but lack data management capabilities such as sub-image extraction from a several hundred GB continuous image. For fast zoom and pan on mosaicked image file sets, many products are available. Access is done through low-level API libraries instead of high-level, model-based query support with internal optimization and without flexible image extraction functionality, such as hyper spectral channel extraction, overlaying, and *ad hoc* thematic coloring. Most important, file-based solutions do not scale well. Relational DBMSs, designed to scale well indeed, traditionally store multidimensional arrays as unstructured BLOBs ("binary large objects") which originally have been introduced by Lorie as "long fields". This technique cannot give any support for operations beyond line-by-line access, something clearly not feasible for large archives. Object-relational database systems (ORDBMSs) allow adding new data types, including access operations, to the server. Arrays, however, are not a data type, but a data type constructor ("template"), parameterized with cell type and dimension. Such templates are not supported by ORDBMSs, hence a separate data type has to be defined for 2-D grayscale ortho images, 2-D hyper spectral MODIS images, 4-D climate models, etc. Furthermore, server internal components are not prepared for the kind of operations occurring in MDD applications, hence important optimization techniques like pipelining and parallelization of array query trees cannot be exploited. Finally, implementing the whole RasDaMan system as an Informix data blade or DB2 extender would mean a complete loss of portability. Hence, we feel that ORDBMSs currently are not an option in array data management, at least for short- and medium-term industrial deployment [11].

Even so it is said that ORDBMS is not recommended as an option in array data management, my project does not deal with any array as it will store attributes with only a single value. Still the best choice in designing my database is to use ORDBMS.

The raster model represents features as a matrix of cells in continuous space. Each cell (or pixel) is a square that represents a specific portion of an area.

- All cells in a raster must be the same size
- Cells are arranged in rows and columns, producing an x,y Cartesian plane, with each cell having a unique x,y value
- All locations are covered by the matrix
- Each layer represents one attribute - attributes are tied to each cell rather than to an area
- Cells with the same value belong to the same zone which do not have to be spatially contiguous (as with most vector features)
- Both integer and floating-point values are supported - continuous data can be represented as either types; categorical data as integer only.

GRASS has a limited internal database that is capable of supporting only a single attribute for each vector object or raster cell category. For projects with more complex data structures, the team of GRASS 5.7 developers has overcome this internal limitation by connecting GRASS to an external database management system such as PostgreSQL (released under the BSD license by The PostgreSQL Global Development Group). This approach parallels ESRI's move to integrate their own data structures with more powerful proprietary databases such as Oracle. In general, connecting GIS to a relational database requires an additional software layer. For ESRI products like ArcInfo, ArcView, and ArcIMS, the functionality for such a gateway is provided by an ArcSDE server. ArcSDE is known to work with a variety of different proprietary databases, including Oracle, Informix, IBM DB2, Sybase, and Microsoft SQL Server. In the case of a GRASS to PostgreSQL connection, the "spatially enabled" functionality is provided by the PostGIS package developed by Refractions Research Inc. (PostGIS is released under the GNU General Public License.) PostGIS allows advanced topological constructs (coverages, surfaces, networks) to be stored, retrieved, and edited in the PostgreSQL object-relational database system. PostGIS works with purely geometric objects, as well. The names of these objects are: point, line, polygon, multipoint, multiline, multipolygon, and geometry collections.

Based on implementation of TerraLib database, global information about a raster layer, such as tile dimension, compression technique, number of rows and columns, number of bands, bounding box, and pixel resolution is stored in a complementary table. I am going to develop my database base on these attributes with some other relevant additional field.

Remote Sensing image data has the unique temporal-spatial characteristics. Hence, a flexible data and storage structure can significantly enhance the system's efficiency and save disk storage (or secondary storage space of the information management system). A suggested structure is shown in Table 2.

Image ID	Spatial Index	Tree Index	Resolution Level	Image Type	Input Time	Compression Format	Image Data	Image size	Image Metadata	Comments

TABLE 2 - A suggested database table structure

2.4 Multi-Resolution or Hierarchical Management of RS Image

"Multi-resolution" is a very important characteristic of RS image management systems because different (or hierarchically organized) levels of RS image detail. In reality the original raw image has the highest resolution and contains the most detailed information. Hence the derived images, or lower level images, have reduced resolution and reduced information content. However, it is the converse when accessing the images, with users typically starting the browsing process from the lowest (least resolution) level, and then increase the image resolution/information gradually. Therefore, multi-resolution management is more practical and more effective than "single-resolution" management.

A "pyramidal structure", see figure 2.3, is the most common approach for multi-resolution. In pyramidal structure each of the derived layers stacks on the top of previous layers. If each higher level has pixels that are exactly twice as wide as the previous, this is called a quad-tree pyramidal structure. Tobler & Chen [1986] describe such a system for coding data for the entire earth's surface. The single node at the top level of the pyramid represents the entire planet, at the 15th level the resolution of the cell is comparable to that of meteorological satellite images; at 26th level the spatial resolution is comparable to most aerial photographs; and at level 30, it is centimeter-scale resolution.

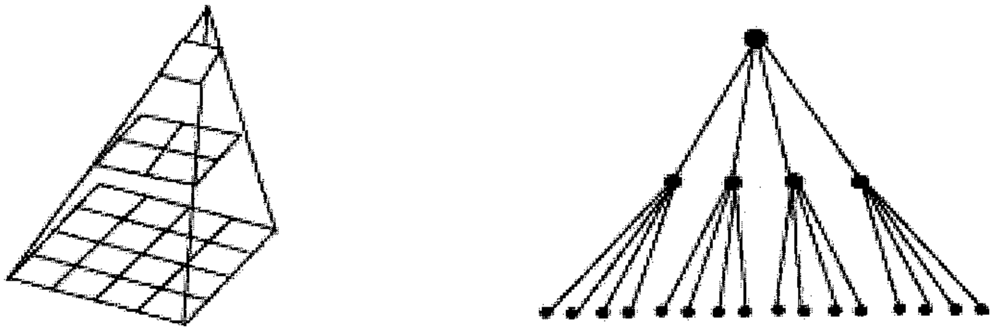


FIGURE 2.3 - Hierarchical RS image structure: pyramid (left) & quad-tree (right)

2.5 Cell Indexing and Searching Mechanism of RS Image

2.5.1 Multi-Dimensional Indices

In the progress of building a RS image system, tessellations and multi-resolution management will improve a system's efficiency significantly. Hence, in order to keep consistent with these two characteristics, a region quad tree is a better choice for indexing. The term 'quad-tree' is used because the underlying data structure is a leveled tree where every leaf node has exactly four descendants. The principle of the region quad-tree is a recursive subdivision of a non-homogeneous square array of pixels into four equal-sized quadrants. The decomposition is applied to each sub-array until there is homogeneity.

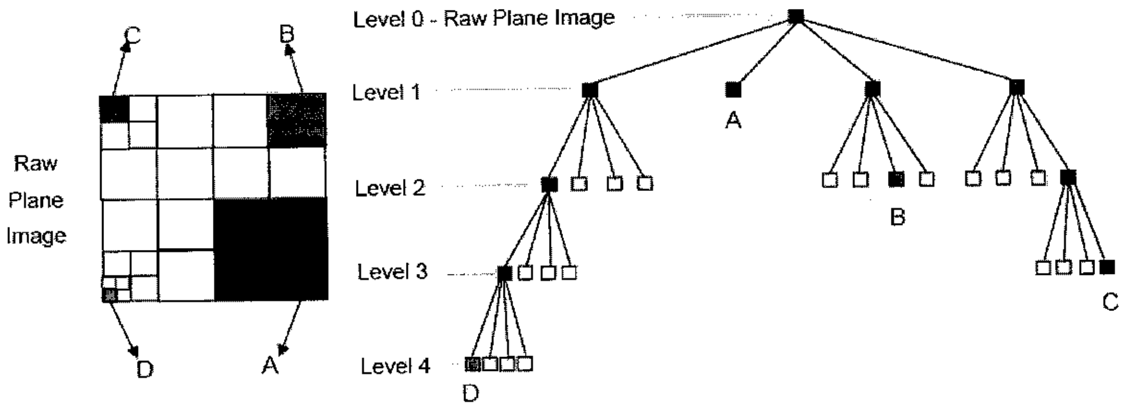


FIGURE 2.4 - Successive subdivisions of the quadrants of RS image

2.5.1.1 Quad tree

The quad tree [14] is used to index 2D space. Each internal node of the tree splits the space into four disjunct subspaces (called NW, NE, SW, SE) according to the axes. Each of these subspaces is split recursively until there is at most one object inside each of them (see Figure 2.5). The quad tree is not balanced and its balance depends on the data distribution and the order of inserting the points.

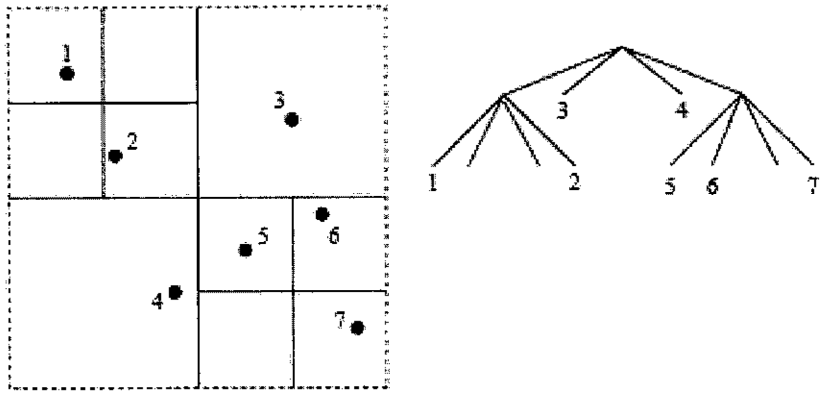


FIGURE 2.5 - Quad Tree

2.5.1.2 MX-Quadtree

In the past three decades, a number of researchers have studied the problem of organizing spatial data, and proposed several multi-dimensional index structures. Currently, a spatial indexing structure that is capable of indexing *regions* covered by images based on their *resolution* level does not exist. MX-quadtree is a spatial index structure for two dimensional points.

MX-quadtree is based on the principle of recursive decomposition; it splits the region into a grid of size 2*k*x2*k* where *k* is the *level of granularity (or level of detail)*. The value of *k* can be freely chosen before constructing the tree, but has to be kept fixed later. Each node is recursively divided into four equal regions. Given a node *N*, its four children represent northwest (N.NW), southwest (N.SW), northeast (N.NE) and southeast (N.SE) quadrants of *N*. All the data objects are stored at the leaf nodes. Figure 2.6 shows a simple representation of the MX-quadtree where the nodes at first level of the tree represent the four quadrants of the region. Each of these quadrants is further divided into four more quadrants, and so on. Points *A,B,C,* and *D* shown in the map on the left are stored based on their position within the region, as shown in the index at right. Insertion and retrieval of an object requires traversal of the tree until the leaf node based on its coordinates.

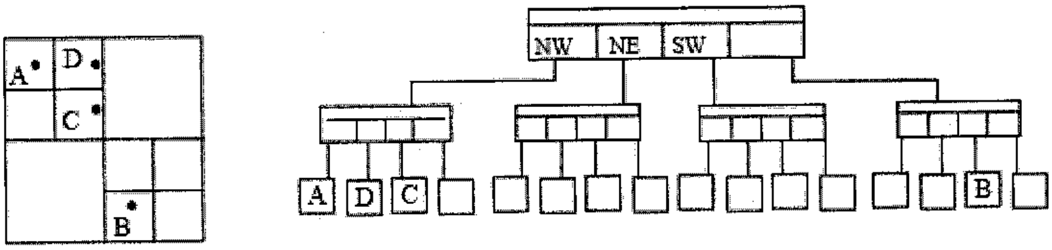


FIGURE 2.6 - MX-quadtree

We will make use of this propriety to extend the MX-quadtree for indexing multi-resolution images.

CHAPTER THREE
METHODOLOGY

A very good database design is needed in order to organize a big amount of data. Thus the database design is crucial in providing a good data management in more effective and systematic way. My project will not deal with any image format conversions. I choose ORDBMS to be applied for my database as it combines the functionalities of object-oriented database modelling and relational database modelling. Nowadays, ORDBMS approach is easy to be put into practice and it is commonly use in developing databases. This approach is mainly for storing satellite images. While all vector layer data will be stored using relational database.

3.1 System Development Life Cycle

In developing the proposed system, I used Waterfall model approach in the system development life cycle. I chose to use this model as documentation are produced at each phase where it fits with other engineering process model. The stages of the model is illustrated in Figure 3.1

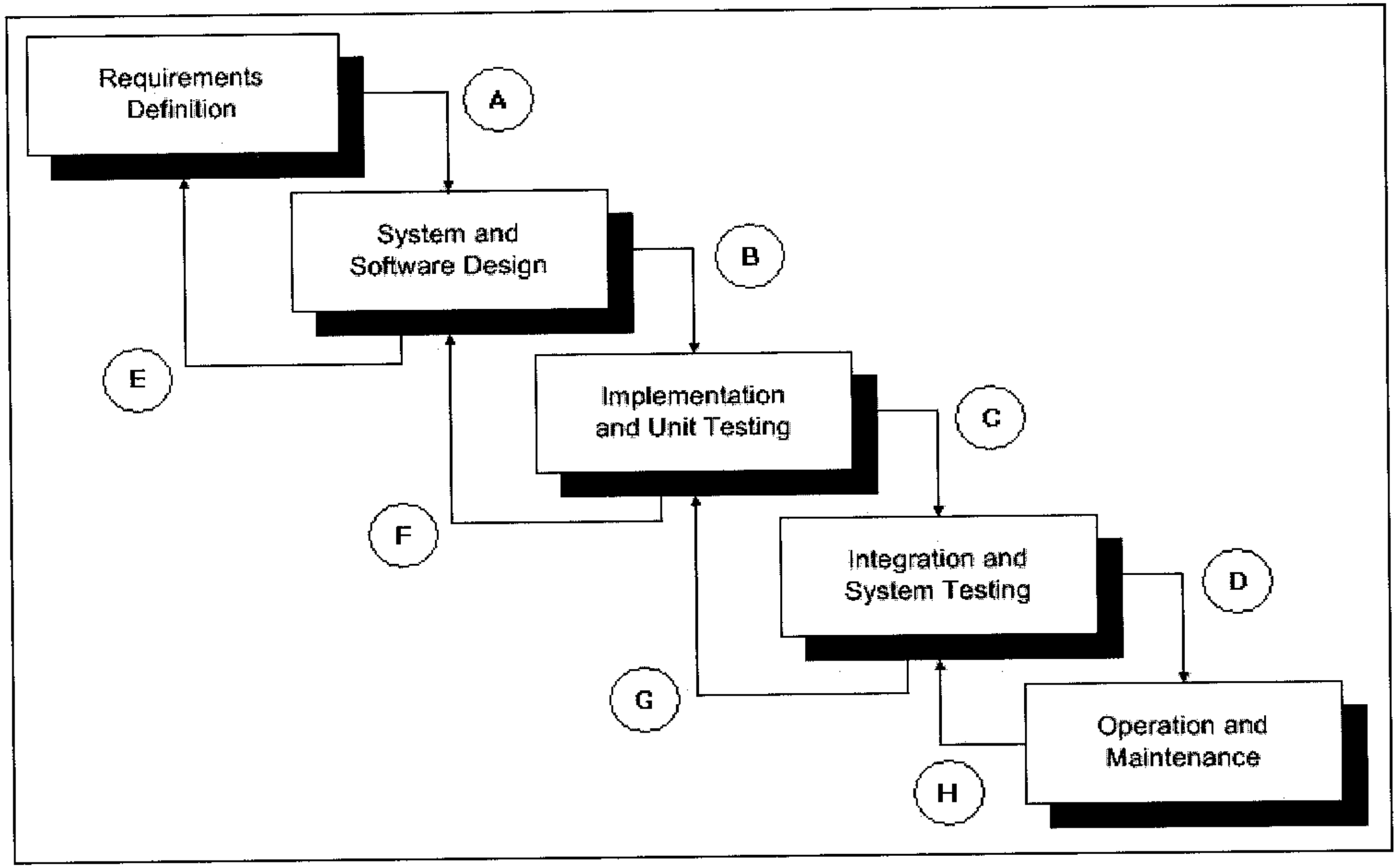

Requirements Definition
A
System and Software Design
B
E
Implementation and Unit Testing
C
F
Integration and System Testing
D
G
Operation and Maintenance
H

FIGURE 3.1 - System Development Life Cycle

The principle stages of the model map onto fundamental development activities which consist of five fundamental stages (A –> D):

A: ***Requirement Analysis and Definition*** – The system's services, constraints and goals are established by consultation with system. They are then defined in detail and serve as a system specification. In this stage, I made thorough analysis on relevant topic to find alternative solution for the specific user requirement and studied current system used in CREM in producing Daily Hotspot report.

B: ***System and Software Design*** – Software design involves identifying and describing the fundamental software system abstraction and their relationship based on user requirements collected before and according to the system architecture and system framework designed. The system design process partitions the requirements of the specific hardware and software systems to be used in this project.

C: ***Implementation and Unit Testing*** – During this stage, required hardware and software for the system is installed and test its functionality in client's workstation. The software design is realised as a set of programs or program units by running the working files on client's localhost. Unit testing involves verifying that each unit meets its specification, also to find error and to make adjustment accordingly.

D: ***Integration and System Testing*** – The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered. However, this system will be developed in my personal computer, thus integration and system testing on client's real workstation is not being covered in this project.

Operation and Maintenance – The system is installed and put into practical use. Maintenance involves correcting errors which are not discovered in earlier stages of the life cycle; improving the implementation of system units and enhancing the system's services as new requirement are discovered. Since there is no real integration and system testing done, the system is not be delivered or to be implemented in my customer's until this project fully finished and approved without any errors.

E: ***Iteration of Requirement Analysis and Definition*** – Adjusted system and software design will be consulted back to the user for comments and suggestions in order to derive a better and efficient solution for this project

F: ***Iteration of System and Software Design*** – If modification is needed during implementation and unit testing phase (e.g., programming language incompatibilities, bugs and errors, inefficient system design), system and software design has to be amended accordingly until it meets the standard requirements.

G & H: ***Iteration of Implementation and Unit Testing*** and ***Operation and Maintenance*** – Yet again, as there is no integration and system testing phase to be run in this project, iteration of this stage is not needed.

In principle, the result of each phase is one or more documents that are approved. The following phase should not start until the previous phase has finished. In practice, these stages overlap and feed information to each other. During design, problems with requirements are identified; during coding design problems are found. The software process is not a simple linear model but involves a sequence of iterations of the development activities.

3.2 Resource Requirements

3.2.1 Hardware

i. Minimum of 500 Megabyte of hard disk storage
ii. Minimum of 256 MB RAM, 512 MB RAM is recommended
iii. 1024 x 768 Display Resolution recommended
iv. Windows XP Home or Professional

3.2.2 Software

i. Ms4w
    a. MapServer
    b. PHP MapScript

c. GMapFactory

d. Apache

ii. Autodesk MapGuide Studio

a. Apache MapGuide

iii. Macromedia Dreamweaver 4.0

iv. PostgreSQL

v. PostGIS

3.3 System Architecture and Framework

There are two types of architectures commonly used in Internet-enabled GIS and database applications: the two-tier and the three-tier client-server architectures as shown in **Appendix 1**. The major difference is whether the client talks directly to the database system or via middle layer (i.e. the server). The two-tier approach allows the client to be generic at the expense of a permanently fixed data structure in the database server or alternatively a flexible database structure at the expense of a non generic client. Neither of these is satisfactory – the database structure must be flexible while allowing for a generic client. The three-tier solution on the other hand does allow for the client to be generic as well as allowing for flexibility in the database server structure and DBMS product as there is a middle tier acting as a mediator between the two. For this reason the three-tier option is required to maintain its generic nature.

In either case the client has the ability to be lightweight, leaving a large portion of the processing to be done on the server side while permitting the client to perform simple tasks such as zooming in and out, panning and identification of selected objects. This means that the client can be run on almost any machine, as the speed of it is not the limiting factor.

The system framework consists of workstation, server and database as shown in a diagram in **Appendix 2**.

Database Server - The database server uses PostgreSQL Database Server storing the spatial data in both raster and vector format. The database consisted of over 1,600 spatial objects or over 20,000 vertices. The size of the spatial and aspatial data in total comes to approximately 450Kb, in relatively good detail. As a result of using vector data it was not necessary to perform all of the spatial operations on the database server. Instead, a portion of these operations, such as selection of objects, zooming, panning and selection of map layers, were performed on the client side. Indexes are created on the spatial and mapping tables makes the database processing time for most types of selection queries will be efficient and independent of the table size.

Server - The server is implemented as a PHP 5.01 application running on an Apache 1.3.6 for Windows web server, browsed in the localhost machine. Two-way communication between the server and the client was achieved by utilizing PHP's implementation of sockets. The primary task of the server is to receive requests from the client, translate them into SQL, request for the database server to perform them and then return the results in an appropriate format to the client. The fact that the client does not communicate to the server in the form of SQL queries means that the client is data independent. This allows the database server to change data structures, data models or even product without visibly impacting the client or user. Since the server acts purely as a mediator between the client and the database, it resulted in being lightweight.

Client - The client process is implemented as a PHPMapScript running in any internet web browser (e.g, Internet Explorer 5) pre-installed with a MapServer application. One important aspect of the client is when it performs a request to the server (and in turn to the database) the retrieved information becomes cached at the client side to remove the need for retrieving the information again at another time. This can be done without any need to worry about "dirty" reads because generally speaking the database is read only. Another reason why caching is desirable is so that the client can perform some spatial processing as mentioned before. The client is the core component of the system and obviously is the component that most time was invested in because of both the need for an intuitive user interface and its need to be efficient.

The bottleneck of the system will, in general, be caused by the speed of the Internet link between the server and the client. For spatial Internet applications, however, the interaction between the server and the backend spatial database can also be very significant as far as processing time is concerned.

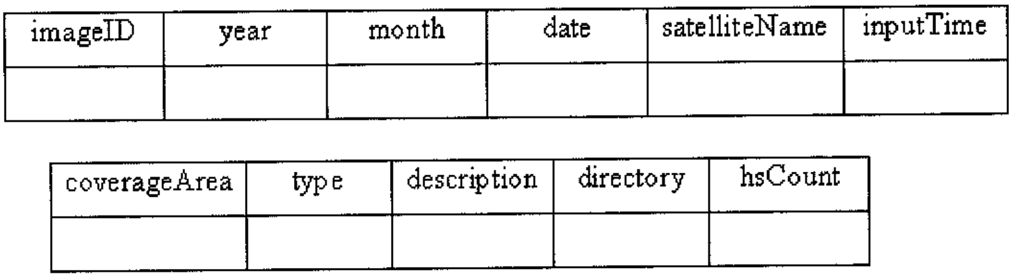
3.4 Database Table Structure

There are another seven vector layers to be displayed together with satellite image (raster layer) in the system. Those additional layers provided which are in vector type are such as:

- Admin – Consists of world international coastal lines
- LatLong – Shows the latitude and longitude lines in the map
- Borneo_PopPlace – Name of all population place in Borneo
- Roads – Existed road data in Borneo
- Rivers – Existed river data in Borneo
- Mountains – Existed mountain data in Borneo
- Hotspots – Hotspot data based on processed satellite images

All vector layers stated above are to be displayed together with raster layer as to enhance the end-user view of this system which are stored in database using relational DBMS which will be linked to satellite images using foreign key.

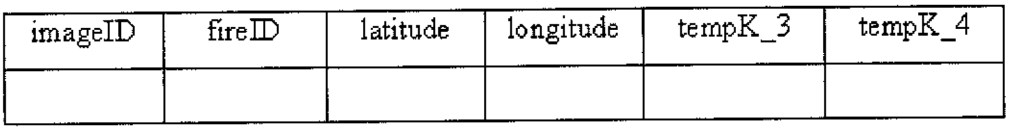
The physical design of database structure for satellite images consists of eleven attributes that store satellite images important data and only six relevant attributes are needed to store hotspot information in the database as in figure 3.2 and figure 3.3 respectively. These two tables are connected and link together using primary key and foreign key method, where the primary key for satellite image entity (***imageID***) is the foreign key for Hotspot entity.



imageID	year	month	date	satelliteName	inputTime

coverageArea	type	description	directory	hsCount

Figure 3.2 - Physical database design for storing satellite images



imageID	fireID	latitude	longitude	tempK_3	tempK_4

Figure 3.3 - Physical database design for storing hotspot information

3.5 Multi-resolution images

As I am using MX-quadtree for indexing multi-resolution images, each satellite images will be divided into 3 levels of multi-resolution tree (level 0 until level 3), which means in each single image, it will be split into 4 equal squares (in level 1), 16 squares (in level 2) and 64 squares (in level 3) as the Figure 3.4a. Below are the examples of images that I have produced.

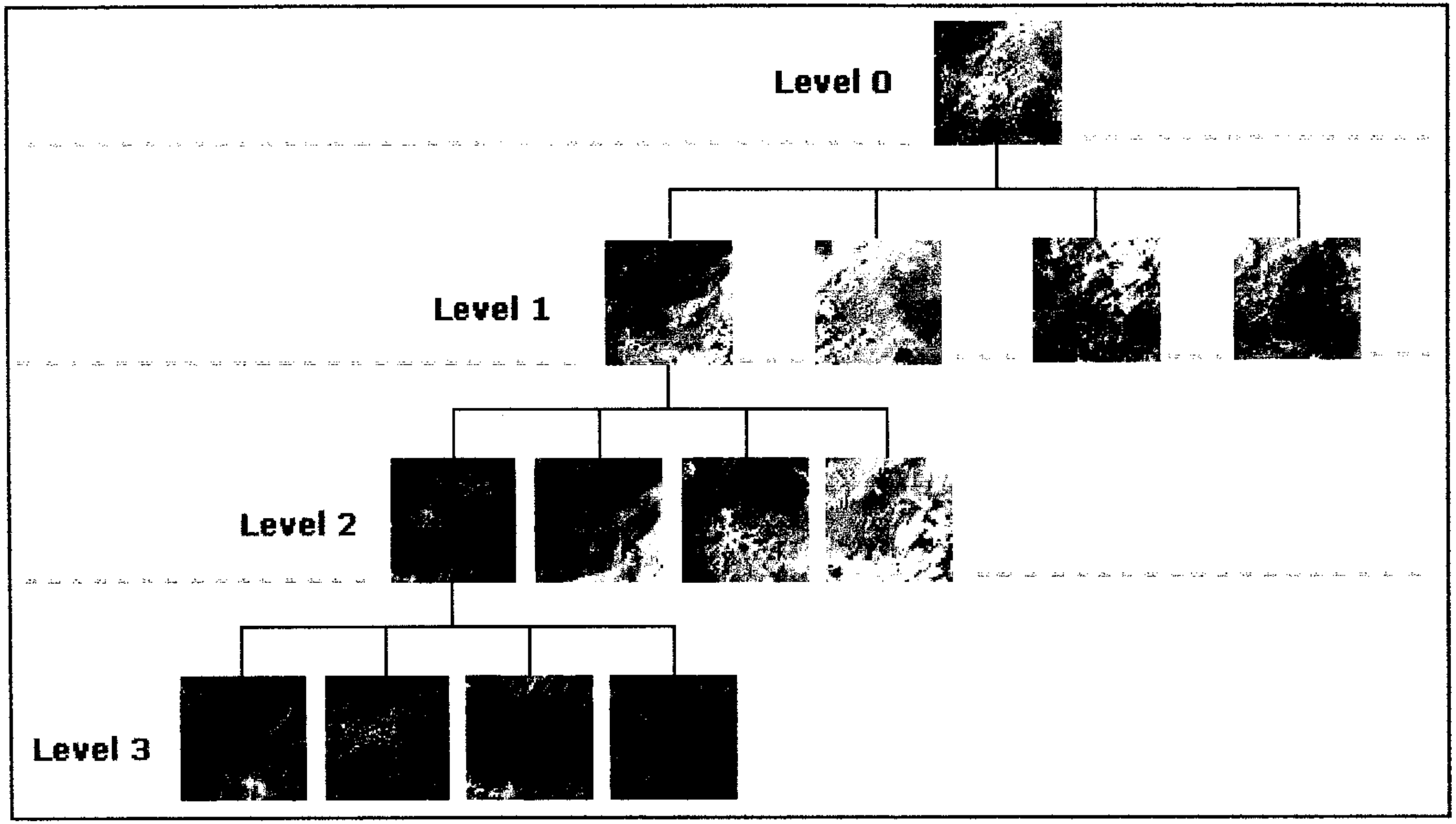


Figure 3.4a - Division of satellite images using MX-quadtree

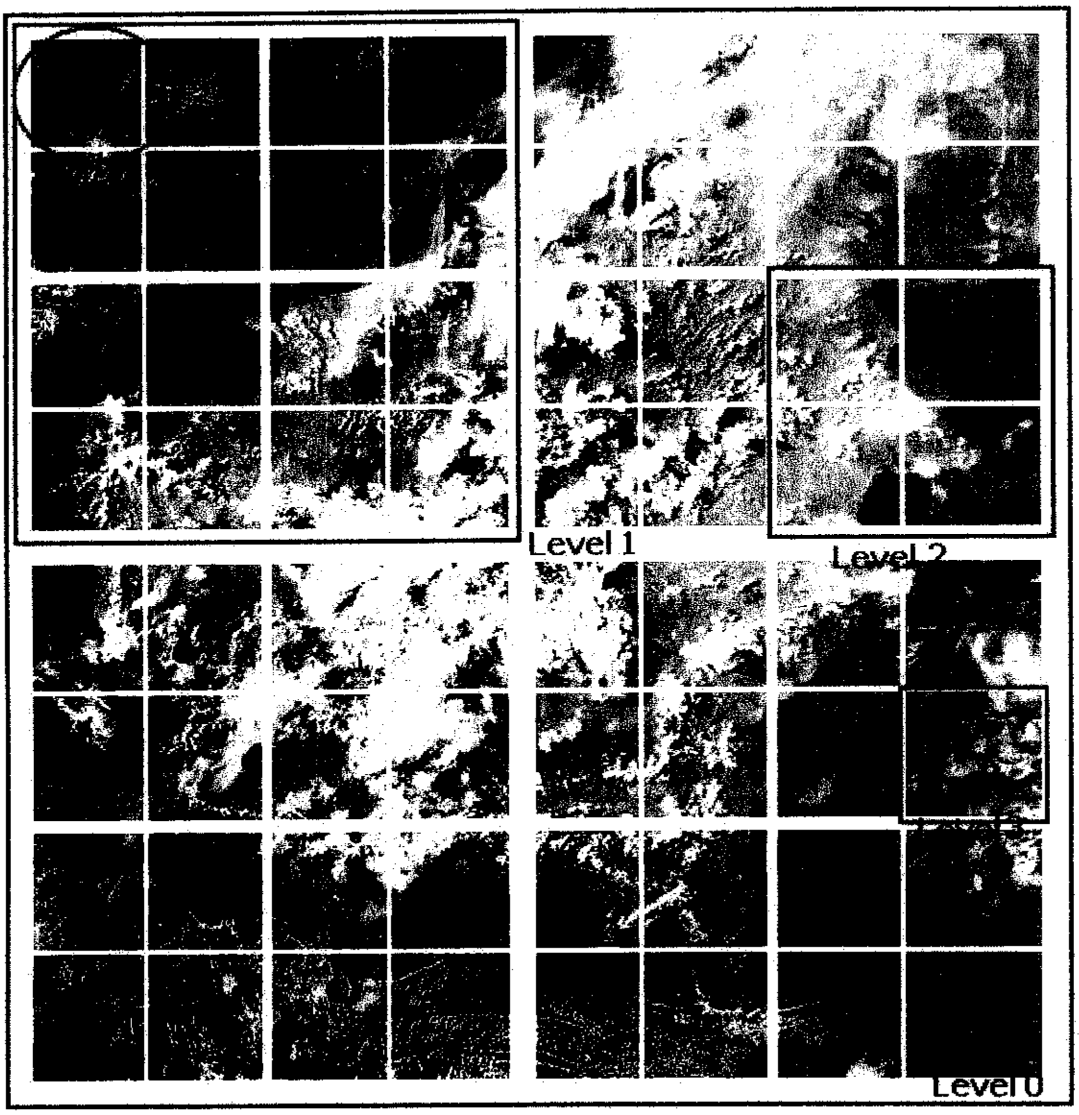


Figure 3.4b - Example of split satellite images

As being circled in Figure 3.4a, the image is in zone A (in level 1) as in figure 3.5.1, within zone 1 area (in level 2) as in figure 3.5.2 and the image in level 3 as in figure 3.5.3. To execute a zooming operation over image, the application decompressed 4 tiles each of the next level. Finally to execute a zooming to a detail level (shown in Figure 3.5.3) the application decompressed another 4 tiles of full resolution.

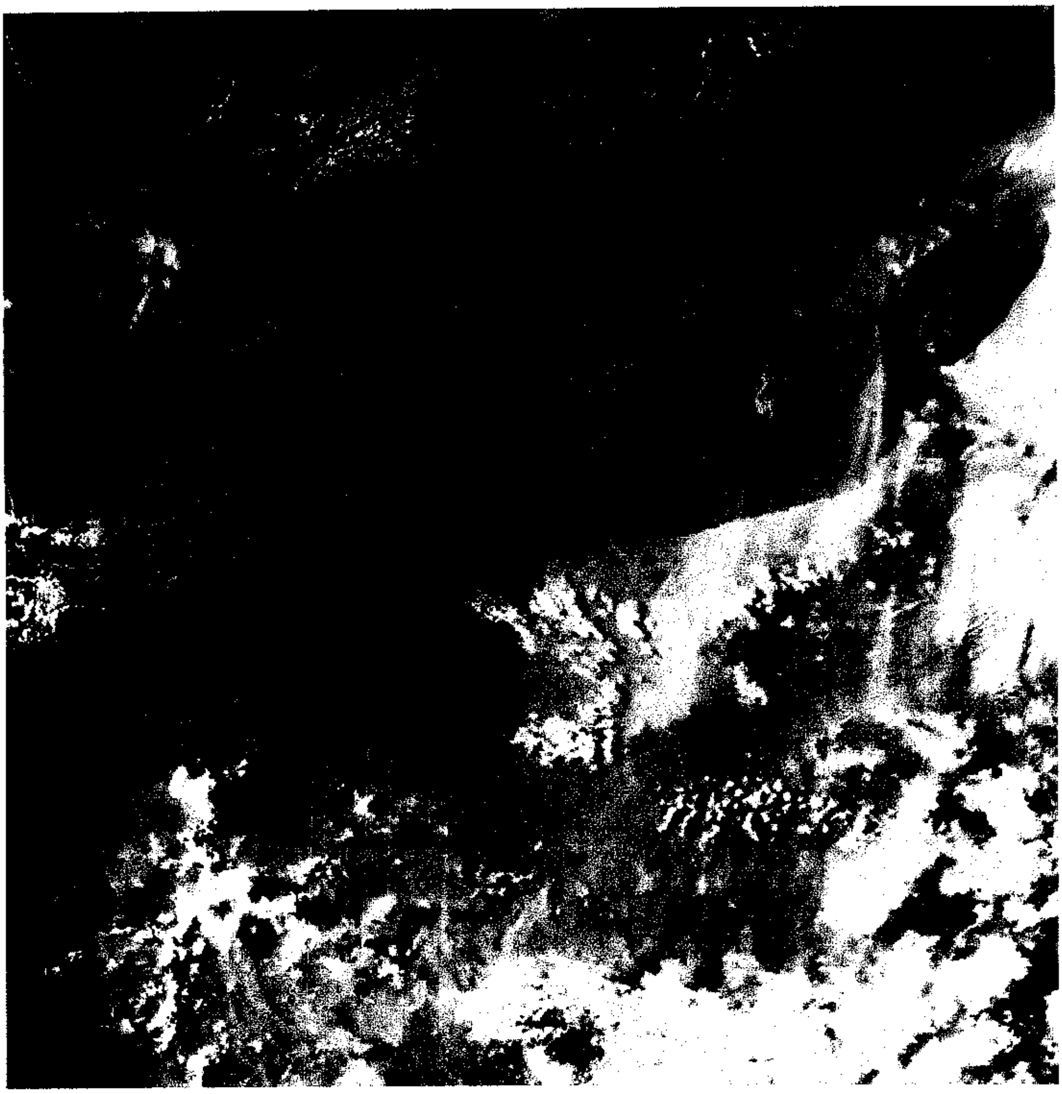


Figure 3.5.1 – Zooming Image in zone A in level 1

Figure 3.5.2 – Zooming detail of Image in zone 1 in level 2

Figure 3.5.3 - Final zooming operation on image within zone A and 1 area

3.6 Website Design

The basic website design for the system is as shown in **Appendix 4**. It consists of three main parts which are:

1. Navigation links on left panel.
2. Satellite images with information
   - Borneo and Peninsular.
   - Hotspot information.
3. Search archive data.
   - Specific data by coverage area, satellite name, day, month or year.

User can choose either to view Borneo or Peninsular Malaysia hotspot map by clicking the navigation link on the left panel. Summary of the hotspots is provided next to the image if the View (in column next to the description of the hotspot map) is clicked as shown in **Appendix 5**.

Search link is accessible in the left panel if user wants to search any particular images from the archive. The webpage design for search and displaying record from database is as shown in **Appendix 6**. There are five list boxes for user to choose in order to search the desired hotspot information by entering value for the combo box provided such as Satellite Name, Date (specific data by day, month or year) and coverage area. With entering values to be queried, it can limit the searching of data and can return results faster. Combination of all or either one of the search criteria will give its result accordingly in the next page once the Search button is clicked.

Meanwhile, Regional Haze Monitoring link shows the latest Regional Satellite Image with Hotspot information for that particular date as shown in **Appendix 7.** Both Peninsular and Borneo satellite data and its hotspot data needs to be combined using other GIS application software in order to come out with the final image to display in this page.

As in **Appendix 8,** it shows the pop-up window for the system to display Hotspot information for particular satellite image.

CHAPTER FOUR
RESULTS AND DISCUSSION

There are mass development of open source software currently available over the internet to be chosen such as PhpGIS, PostGIS, PostgreSQL, Mapsever, Maplab, Ka-map, Chameleon, GmapFactory, GRASS, Autodesk MapGuide Studio, MapGuide and many other more to come in future as I mention it now. I made comparison of some available softwares before hand to find out the best software suit for the proposed system (mainly MapServer using GMapFactory available functions and Autodesk MapGuide Studio) that I had installed in my local machine before to make sure that it can function as expected.

The MapServer application of OpenSource GIS software is the most suitable application for the purpose this project. Together with GMapFactory application, manipulating both vector and raster data is so easy with existence of useful functions offered in MapServer. In other words, it supports for tiled raster and vector data (display only). User does not have to split raster image using any multi-resolution image because the zoom function and the tile-indexing for each raster image is ready-made in MapServer as is has quadtree spatial indexing for shapefiles.

However there are some configurations need to be done before the data can be successfully used in MapServer application. Due to limited time constraint, limited skill on programming language (PHPMapScript) and unfamiliar functions provided in the application, I faced big problems in continuing my project development. Even though there are many tutorials about MapServer application available in the internet, it is so difficult for me to understand and implement the solutions provided to my project. Until the end of expected date for system development, I still did not manage to solve the problem to display the map (vector data or even raster) using MapServer application. Complicated functions with limited explanations make my project wedged and left unsolved. User has to know georeference coordinates for the data (vector and raster) in order to be displayed in MapServer application, otherwise the data will not be displayed in the application thus make it can not be used at all. Projection setting is also needed in order to display the raster data correctly according to the world map. The biggest limitation of this MapServer application is it only supports 8-bit raster format type such as TIFF/GeoTIFF, GIF, PNG, ERDAS, JPEG and EPPL7. The raster data given by CREM are in .bmp format which is not supported by MapServer application thus makes it not possible to be used.

Another open source GIS application that I have tried, Autodesk MapGuide Studio also has the similar problem where inconsistent Apache Mapserver Guide connection in workstation made my project difficult to develop. Although it is a very good newly OpenSource application (upgraded version of MapServer application), it does not fulfill a compulsory requirement for this project that is the ability to connect to Object-relation database (PostgreSQL). Furthermore, its database configuration and display setting are complicated enough that I did not have time to do further analysis on that particular matter.

On the other hand, I have successfully made connection on PostgreSQL database to MapServer application which is the core purpose of this project. The database to store raster images has been created but the relationship to other vector data has not been done yet. PostgreSQL is a complicated object-relational DBMS where it needs me to make thorough study on its special SQL syntaxes before I can figure out how to fully use the functions available.

As the output for the project, I have developed the system using PHP programming language together with creation of LAC database using MySQL. As PHP language does not support spatial data, it can only manipulate raster images in any format (which in my project, I have combined both raster and vector data to come out with a single .bmp image). I created two tables for the purpose of this project for image and hotspot entities. It has one-to-many relationship that the tables are connected using primary key in image table as a foreign key in hotspot table.

Information displayed in the system is based on whatever data stored in database by the system administrator. There is no such GIS image processing or any other functions on the raster images can be done in the system.

For this project, I did comparison on the efficiency of linking entities between object-relational DBMS and relation DBMS. I found out that object-relational uses inheritance method and some other ways (nested table, OID method and etc.) in order to link to other entities while relational DBMS only uses primary and foreign key to do the relationship between tables which is easier and less complicated. Due to behind project schedule, I decided to chose to create my database using relational DBMS instead rather than object-relational. The data are retrieved by writing SQL syntax to call both tables which are link by the foreign key.

The system consists of a GIS database system for storing maps, raster images, and hotspots information. Since no other GIS application software can successfully configured to customize my system, thus the data has been manually converted to fully-raster format using other GIS software such as ArcView or Autodesk Map in order to display it in my system thus make it less attractive for the user-view.

Possible scenario of operations happen in the system are storing GIS maps, hotspot images and its relevant information; and retrieving selected images from archive. Given that the system developed using PHP language, it limits the data to be static raster as it is, which mean there is no function build to zoom in the data as it does not provided by PHP unless the system developer integrate it with JavaScript.

CHAPTER FIVE
CONCLUSION AND RECOMMENDATION

Data input is a major bottleneck in any application of GIS technology. It gives big effects on system development. As for my project, the data that I have are not being supported by some GIS application software thus make it impossible to manipulate and develop custom functions for it. Based on my research for this project, GIS field in open source still need to find some alternative tools to support more spatial data of various types into digital format.

It is also my finding to know the importance of database design to provide quality information regardless what kind of DBMS being used. This paper shows that it is not possible to develop a proper organizes database despite object-relational or relational database management system. The system developer has to carefully design the database in order to successfully retrieve desired information from database and write SQL syntax correctly.

Here I present various methods and things to be considered as to store and displaying satellite images on a web-application ranges form the data itself, database design and the web page design flows.

Even though there are many open source GIS application being developed by programmers until today, one must has its own advantages and also disadvantages as I discussed in the previous chapter before. As for MapServer application, it can be enhanced by supporting more wide range of data format especially the raster data in higher bit format. Displaying both raster and vector data with less complicated configuration will make MapServer application the best open source software for all system developers even for novice users.

A sample of multi-resolution satellite image consumes a lot of space because there are many images being produced. The higher level of multi-resolution image, the higher number amount of images needed to be stored. This would be a hassle for big size satellite images, to be split into various levels of multi-resolution and to stored it into database.

For providing effective and efficient means to serve access requests, a uniform indexing structure, called *MX-quadtree,* which is capable of indexing multiresolution satellite imagery. The web application will have basic functionalities such as *view* and *zoom-in, zoom-out* as to give power to the user to choose specific location on hotspots map to be viewed. Furthermore, I intend to do a research on how to display raster and vector image simultaneously to enhance the presentation of hotspot maps in the web-application.

The initial intention for this project is to customize any GIS open source application software (MapServer configuration by using GMapFactory application), some how towards the end of system development phase, the configuration problems can not be solved – my current raster data is not supported by the application and the vector data can not be displayed as I could not find the solution for projection and some other hardly to understand MapServer settings. As for this project, I have developed my own application using PHP and MySQL database which can only display Hotspot images (raster format) not together with the other seven vector data.

As for recommendation for future, I would like to extend the system functionalities by having functionality to zoom (zoom-in or zoom-out) raster image. And to provide more easily-meaningful data (population names instead of latitude and longitude coordinates) to be displayed in the system to users instead of static meaningless data.

REFERENCE

[1] Agin J., 1979, *In: Proc. of the DARPA Image Understanding Workshop,* 66-71

[2] McKeown D., Harvey W., McDermott J., 1985, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(5), 570

[3] Nicolin B., Gabler R., 1987, *IEEE Transactions on Geoscience and Remote Sensing*, 25(3), 317

[4] Matsuyama T., Hwang V., 1990, SIGMA: *A Knowledge-Based Aerial Image Understanding System*, Advances in Computer Vision and Machine Intelligence - Plenum Press

[5] Quint F., Sties M., 1995, In: *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, 307-316

[6] Liedtke C., Buckner J., Grau O., Growe S., Tonjes R., 1997, In: *3rd Int. Airborne Remote Sensing Conference and Exhibition,* vol. I, 308-315, Copenhagen

[7] Koch H., Pakzad K., Tonjes R., 1997, In: *SMATI97, workshop on Semantic Modeling of Topographic Information from Images and Maps*, Bonn, Germany

[8] Koelma D., Smeulders A., 1999, In: *VISUAL99, Third Int. Conf. on Visual Information Systems*, Amsterdam, The Netherlands

[9] Nevatia, R. and K. Price, *Automatic and Interactive Modeling of Buildings in Urban Environments from Aerial Images* . IEEE ICIP 2002, 2002. **III**: p. 525-528.

[10] Gilberto Camara, *SPRING: INTEGRATING REMOTE SENSING AND GIS BY OBJECTORIENTED DATA MODELLING*
National Institute for Space Research (INPE), Brazil

[11] Peter Baumann, *Web-enabled Raster GIS Services for Large Image and Map Databases*
Active Knowledge GmbH, Kirchenstr. 88, D-81675 Munich, Germany

[12] Lubia Vinhas, Ricardo Cartaxo Modesto De Souza, *IMAGE DATA HANDLING IN SPATIAL DATABASES*
Instituto Nacional de Pesquisas Espaciais - INPE

[13] Ming-Hsiang Tsou, *An Operational Metadata Framework for Searching, Indexing, and Retrieving Distributed Geographic Information Services on the Internet*
Department of Geography, San Diego State University, CA, USA
<http://typhoon.sdsu.edu/tsou/index.html>

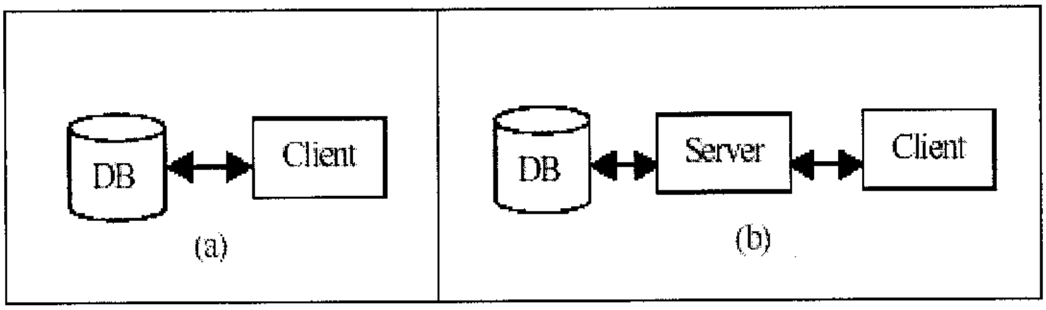
[14] Finkel R., Bentley J.: *Quad Trees: A Data Structure for Retrieval on Multiple Keys.* Acta Informatica, Vol. 4, No. 1, 1974, pp. 1-9.

[15] Bentley J.: *Multidimensional Binary Search Trees used for Associative Searching.* Communications of ACM, 18, 9, Sep. 1975, pp. 509-517.

[16] Beckmann N., Kriegel H. P., Schneider R. and Seeger B.: *The R*-tree: an efficient and robust access method for points and rectangles* Proceedings of the 1990 ACM SIGMOD international conference on Management of data, 1990, Atlantic City, NJ USA, pp. 322-331.

[17] Sellis T. K., Roussopoulos R., Faloutsos C.: *The R+-Tree: A Dynamic Index for Multi-Dimensional Objects.* Proc. 13th International Conference on Very Large Data Bases, (VLDB'87), Brighton, England, 1987, pp. 507-518, Morgan Kaufmann, ISBN 0-934613-46-3.

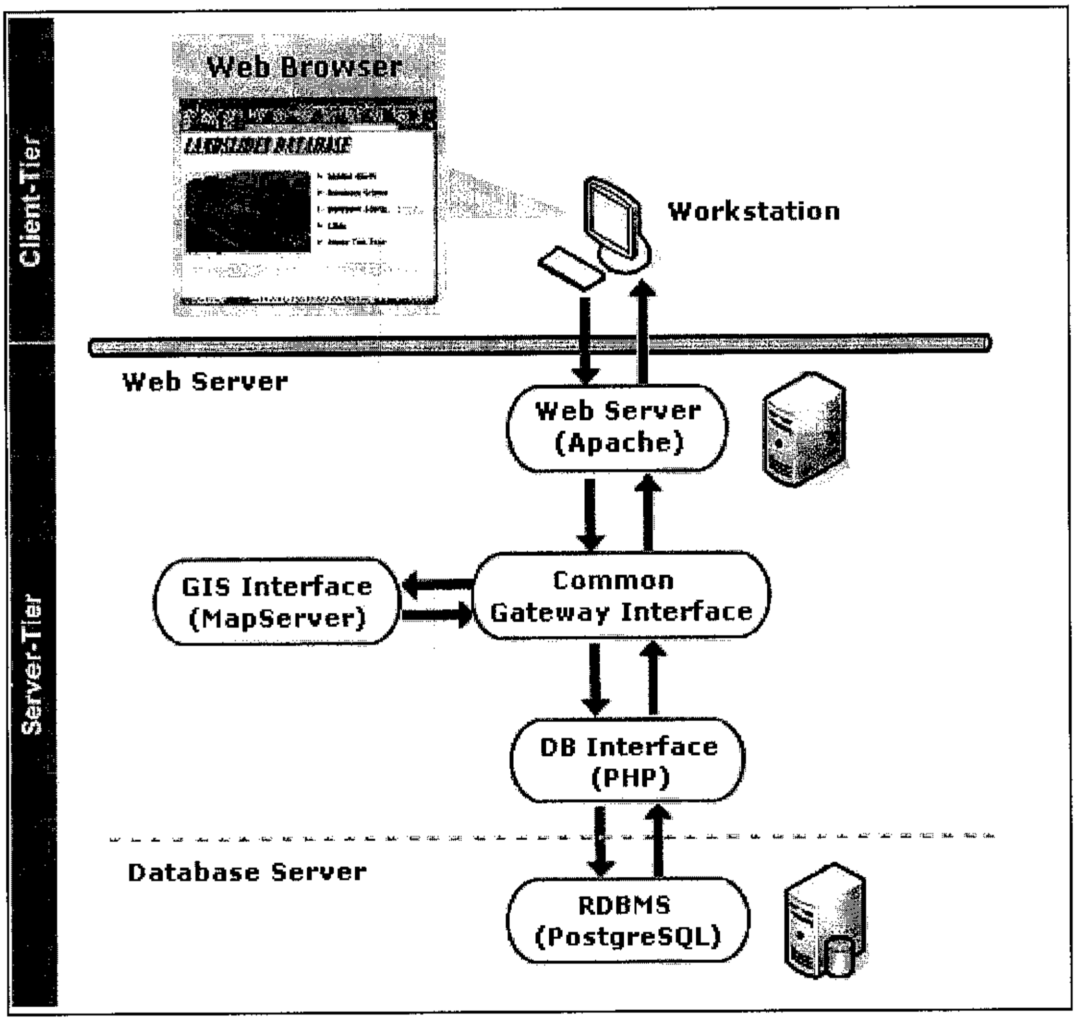
APPENDICES



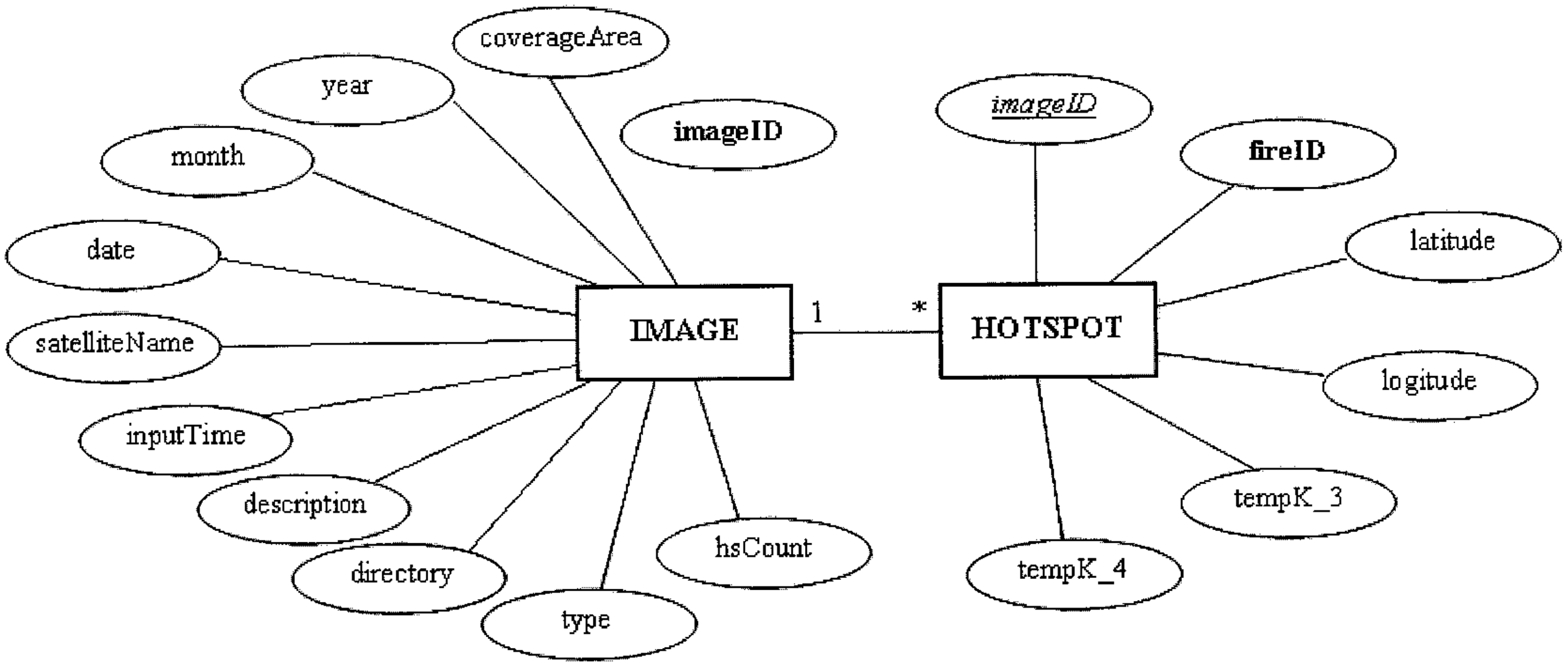
Appendix 1 – Two types of system architectures

a) Two-tier architectures

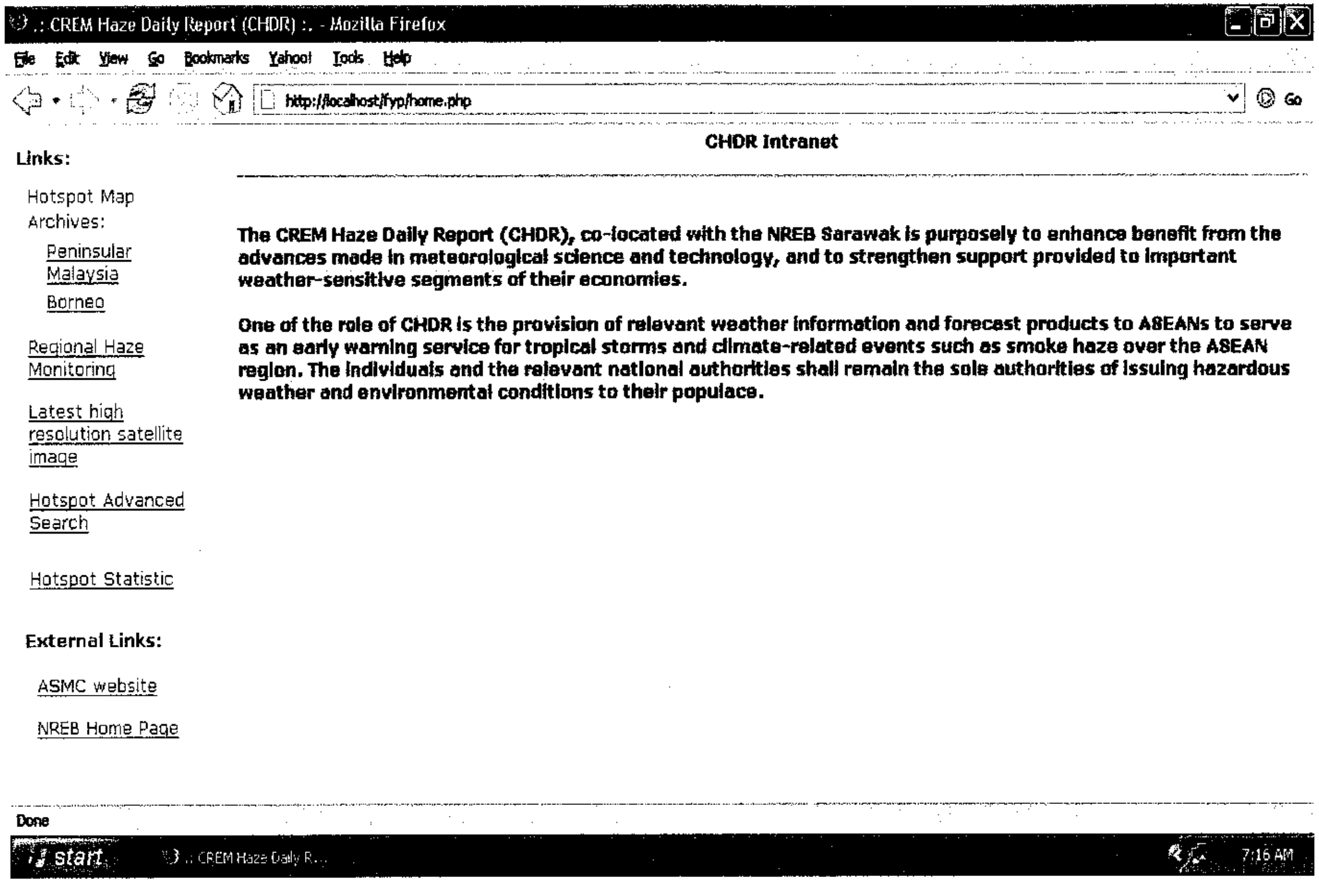
b) Three-Tier client-server architectures



Appendix 2 – System Framework



Appendix 3 – ER Diagram for Satellite Image and Hotspot information

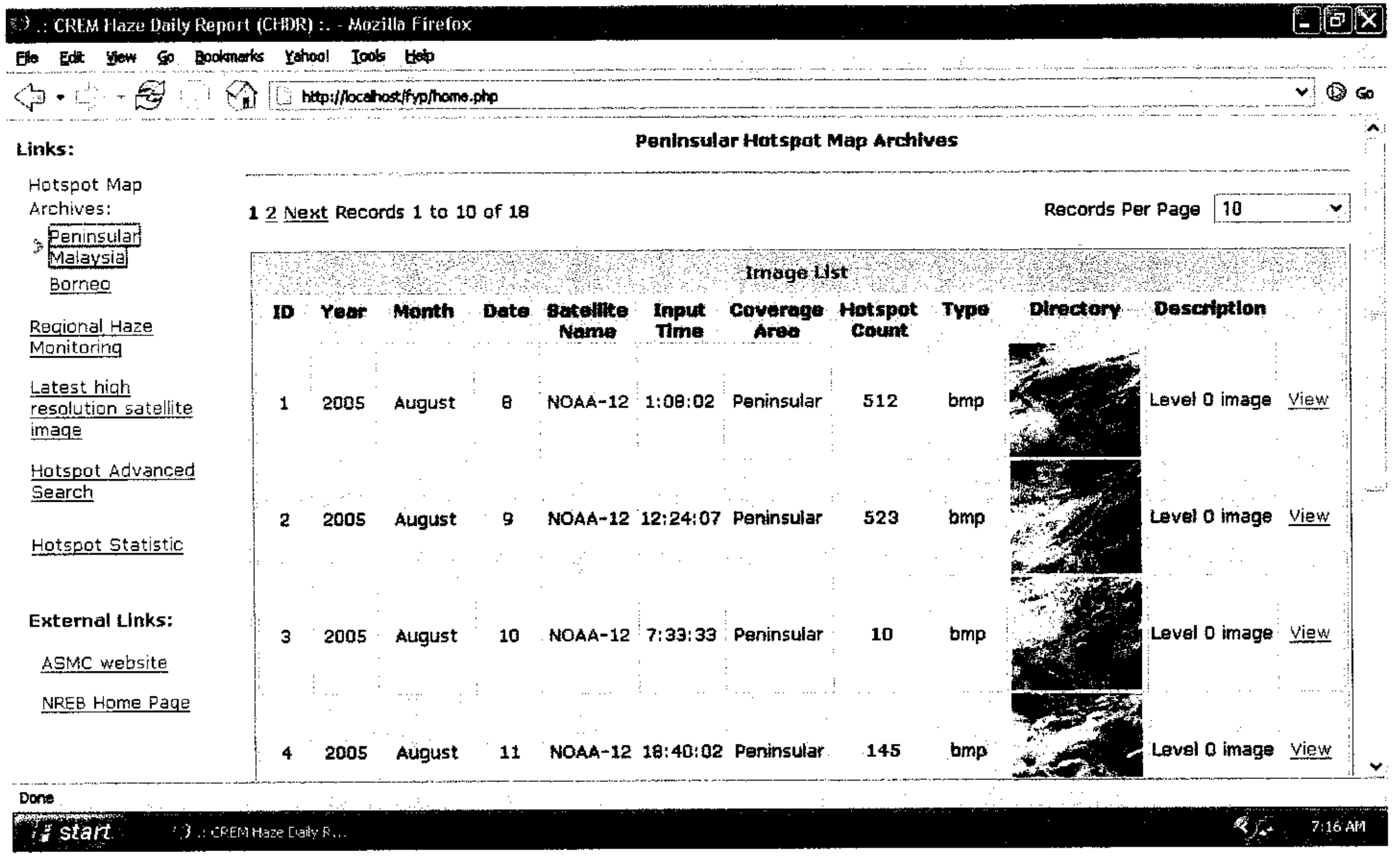


CHDR Intranet

The CREM Haze Daily Report (CHDR), co-located with the NREB Sarawak is purposely to enhance benefit from the advances made in meteorological science and technology, and to strengthen support provided to important weather-sensitive segments of their economies.

One of the role of CHDR is the provision of relevant weather information and forecast products to ASEANs to serve as an early warning service for tropical storms and climate-related events such as smoke haze over the ASEAN region. The individuals and the relevant national authorities shall remain the sole authorities of issuing hazardous weather and environmental conditions to their populace.

Appendix 4 – CHDR System Main page



.: CREM Haze Daily Report (CHDR) :. - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/fyp/home.php
Go
Done
start
.: CREM Haze Daily R...
7:16 AM

Links:

Hotspot Map Archives:

- Peninsular Malaysia
- Borneo

Regional Haze Monitoring

Latest high resolution satellite image

Hotspot Advanced Search

Hotspot Statistic

External Links:

- ASMC website
- NREB Home Page

Peninsular Hotspot Map Archives

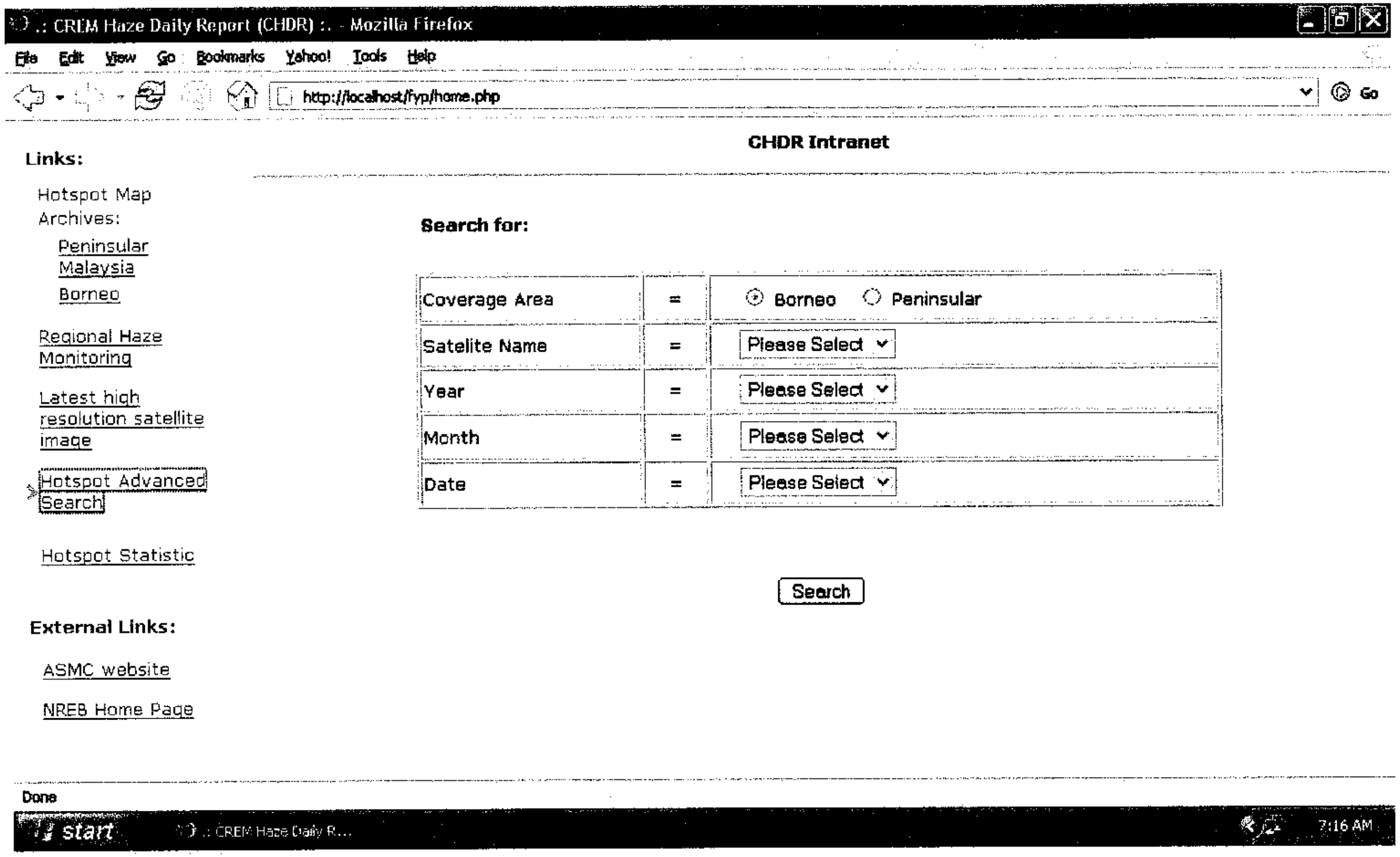
1 2 Next Records 1 to 10 of 18

Records Per Page 10

Image List

ID	Year	Month	Date	Satellite Name	Input Time	Coverage Area	Hotspot Count	Type	Directory	Description	
1	2005	August	8	NOAA-12	1:08:02	Peninsular	512	bmp		Level 0 image	View
2	2005	August	9	NOAA-12	12:24:07	Peninsular	523	bmp		Level 0 image	View
3	2005	August	10	NOAA-12	7:33:33	Peninsular	10	bmp		Level 0 image	View
4	2005	August	11	NOAA-12	18:40:02	Peninsular	145	bmp		Level 0 image	View

Appendix 5 – Hotspot Map archive (Peninsular data)


.: CREM Haze Daily Report (CHDR) :. - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/fyp/home.php
Go
Done
start
.: CREM Haze Daily R...
7:16 AM

CHDR Intranet

Links:

Hotspot Map
Archives:
Peninsular Malaysia
Borneo

Regional Haze Monitoring

Latest high resolution satellite image

Hotspot Advanced Search

Hotspot Statistic

External Links:

ASMC website

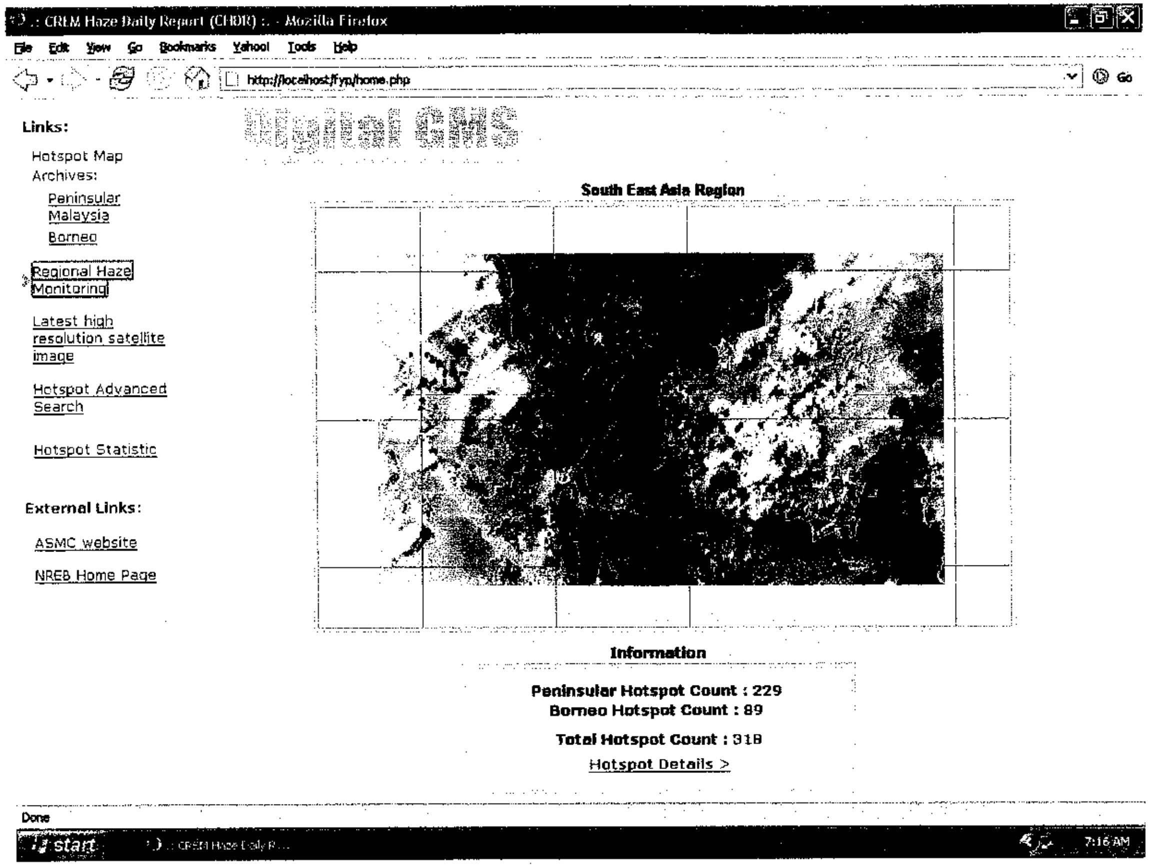
NREB Home Page

Search for:

Coverage Area	=	● Borneo ○ Peninsular
Satelite Name	=	Please Select
Year	=	Please Select
Month	=	Please Select
Date	=	Please Select

Search

Appendix 6 – Search Hotspot data in archive



.: CREM Haze Daily Report (CHDR) :. - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/fyp/home.php
Go
Digital GMS
Done
start
.: CREM Haze Daily R...
7:16 AM

Links:

Hotspot Map
Archives:
Peninsular Malaysia
Borneo

Regional Haze Monitoring

Latest high resolution satellite image

Hotspot Advanced Search

Hotspot Statistic

External Links:

ASMC website

NREB Home Page

South East Asia Region

Information

Peninsular Hotspot Count : 229
Borneo Hotspot Count : 89

Total Hotspot Count : 318

Hotspot Details >

Appendix 7 – Latest Regional Satellite Image with Hotspot information

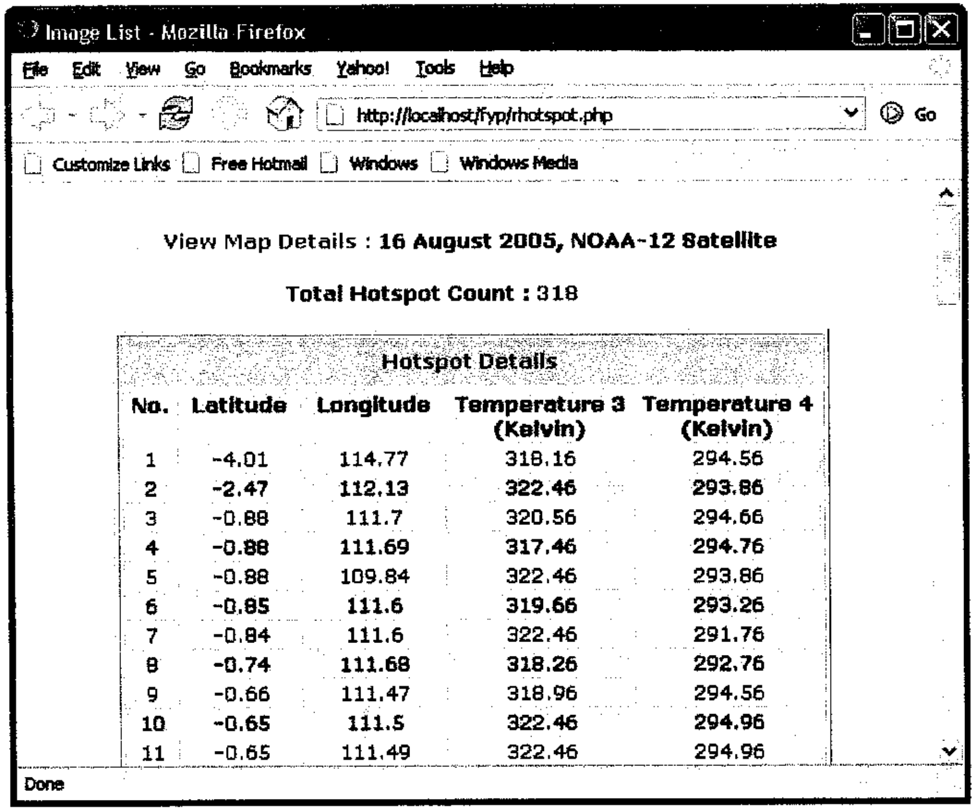


Image List - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://localhost/fyp/rhotspot.php
Go
Customize Links Free Hotmail Windows Windows Media
Done

View Map Details : **16 August 2005, NOAA-12 Satellite**

Total Hotspot Count : 318

Hotspot Details				
No.	**Latitude**	**Longitude**	**Temperature 3 (Kelvin)**	**Temperature 4 (Kelvin)**
1	-4.01	114.77	318.16	294.56
2	-2.47	112.13	322.46	293.86
3	-0.88	111.7	320.56	294.66
4	-0.88	111.69	317.46	294.76
5	-0.88	109.84	322.46	293.86
6	-0.85	111.6	319.66	293.26
7	-0.84	111.6	322.46	291.76
8	-0.74	111.68	318.26	292.76
9	-0.66	111.47	318.96	294.56
10	-0.65	111.5	322.46	294.96
11	-0.65	111.49	322.46	294.96

Appendix 8 – Hotspot information for particular satellite image