Hierarchical Land Cover Information Retrieval in Object-Oriented Remote Sensing Image Databases with Native Queries

Jiang Li
Department of Computer Science and Information Technology,
Austin Peay State University, Clarksville, TN 37044
Tel: 1-931-221-7828
Email: lij@apsu.edu

ABSTRACT
Classification and change detection of land cover types in the remotely sensed images is one of the major applications in remote sensing. This paper presents a hierarchical framework for land cover information storage and retrieval from object-oriented (OO) remote sensing image databases. Multi-spectral (band) remotely sensed images are classified by an optimized *k*-means clustering algorithm. The land cover maps are then decomposed and indexed with region quad-tree data structure stored in an OO database. Native queries (NQs), which use the semantics of the OO programming language for query composition, are developed to retrieve land cover distribution information and detect the changes of each land cover type at multi levels. A prototype system was implemented and the experiments were conducted on a time series Landsat Thematic Mapper (TM) images. The results show the effectiveness of the framework and the potentials in other remote sensing applications like urban planning and drought monitoring.

Categories and Subject Descriptors
J.J.2 [**Physical Sciences and Engineering**]: *Earth and atmospheric sciences*

General Terms
Algorithms, Management, Design, Experimentation.

Keywords
information retrieval, change detection, clustering, object-oriented databases, remote sensing.

1. INTRODUCTION
Land cover / land use classification and change detection in the remotely sensed images is one of the most important applications in remote sensing. Many algorithms have been proposed for multi-spectral image classification. Supervised classification such as Bayesian [1] and Support Vector Machines [2] can be applied when training data and/or ground truth data can be obtained from the study area. Unsupervised classification techniques like clustering [3] or mathematical morphology [4] may be used to find similar groups in the data, each representing a class. A label will then be assigned to each class by domain experts during the post-classification processing.

ACMSE 2007, March 23-24, 2007, Winston-Salem, North Carolina, USA ©Copyright 2007 ACM 978-1-59593-629-5/07/0003...$5.00

K-means clustering is a widely used unsupervised algorithm applied in pattern recognition due to its properties of local-minimum convergence [5] and implementation simplicity. However, it suffers some intrinsic deficiencies. First, it is sensitive to initial starting conditions, i.e., the clusters are fully deterministic given the randomly or arbitrarily chosen initial centers. Second, the number of clusters has to be provided as a parameter, which assumes *a priori* knowledge about the data is available. Third, computation is expensive as it requires multiple data scans to achieve convergence. Although a universal solution does not exist, various approaches have been proposed as partial remedies. Speed can be improved by embedding the data set in a multi-resolution *kd*-tree and storing sufficient statistics at its nodes [6]. A new step is introduced to the *k*-means clustering process to iteratively update variable weights based on the current partition of data and a formula for weight calculation [7].

The size of remotely sensed images is usually very large compared to the images used in other domains. For instance, a 7-band Landsat TM image consists of near 25 million pixels, which consumes approximately 250M bytes storage. Given the resolution of the Landsat TM image is 30 meters per pixel, a full scale image covers scene over 10,000 square miles. However, users are usually interested in different levels of details. Hierarchical management that facilitates the query on the image database and save time storing or loading images, therefore, becomes an important characteristic of remote sensing image information retrieval systems. A region quad-tree [8] structure is used for multi-level image indexing in this study.

Image objects with region quad-tree indexes will need to be stored in databases. Traditional relational database model with object-relational mapper typically stores image data as a binary blob field with associated properties as separate fields in a table. Objects created with OO programming languages must be mapped to entities in databases. This mapping mechanism increases the management overhead and makes the debugging process difficult. In this study, we adopt a pure object-oriented database (OODB) model, which accesses to data through relationships stored within the data themselves. Meanwhile, it supports native queries [9] that use the semantics of the OO programming language for query composition.

The rest of the paper is organized as follows. Section 2 discusses classifying multi-spectral images into land cover maps using an optimized *k*-means clustering approach. Section 3 describes the hierarchical management with region quad-tree indexing. OO database diagram and native queries are presented in section 4. Section 5 discusses experiments of land cover distribution information retrieval and change detection of each land cover type at multi levels. Section 6 concludes with proposals for future work.

2. LAND COVER CLSSIFICATION BY OPTIMIZED *K*-MEANS CLUSTERING

For *k*-means clustering, a cluster validity measure is proposed in [10] to quickly estimate number of clusters. Bradley and Fayyad [11] discuss an approach to refine the selection of starting centers through repeated sub-sampling and smoothing. Pena and Larranaga [12] propose an empirical comparison of several initialization methods. To generate land cover map, we apply the optimized *k*-means clustering procedure [13] that combines the initial centers refinement algorithm with the cluster validation measure.

Because the 6th band in a Landsat TM image is generated at infrared wavelength with 120-meter resolution, it is not used in the classification procedure. Suppose N is the total number of pixels, K is the number of clusters, $\mathbf{x}$ is the vector $[b_1, b_2, b_3, b_4, b_5, b_7]$ of pixel values that represents the spectral information in each band, and $\mathbf{m_i}$ is the center of cluster C_i, the intra-cluster distance is defined as the average of the sum of distance between each sample and its cluster center

$$M_{intra} = \frac{1}{N}\sum_{i=1}^{K}\sum_{\mathbf{x}\in C_i}\left\|\mathbf{x}-\mathbf{m}_i\right\|^2 \qquad (1)$$

The inter-cluster distance is defined as the minimum distance between any two cluster centers

$$M_{inter} = \min\left\|\mathrm{m}_i-\mathrm{m}_j\right\|^2 \qquad (2)$$
$$i=1,2,\ldots,K-1,\quad j=i+1,\ldots K$$

The clustering process is to find clusters with the minimal intra-cluster distance and maximal inter-cluster distance, i.e., the smallest validity measure, which is defined as the ratio of these two distances M_{intra} / M_{inter}.

The initial center refinement algorithm is briefly summarized as follows. J small sub-samples S_i, i = 1, 2,…, J, are randomly selected and *k*-means clustering produces solutions CM_i, which are estimates of the true cluster centers. The algorithm checks the solution at termination for empty clusters and sets the initial estimates of the empty cluster centers to data points that are farthest from their assigned cluster center. A distortion value is computed as the sum of square distances of each data point to its nearest center. The centers with minimal distortion over each sub-sample set are chosen as the refined initial centers.

Assumes K_{max} is the upper limit of the total number of clusters, to find the optimal number of clusters, for each integer K where $2 \le K \le K_{max}$, the optimal K_{opt} is selected by clustering with a minimum validity measure. The optimization procedure runs the cluster validation algorithm on each sub-sample data set S_i, and takes the mean value of each $K_{opt\text{-}i}$ as the optimal number of clusters K_{opt}. This value is then used as one of the input parameters of the refinement algorithm to find the optimized starting centers.

Figure 1 (a) and (b) are examples of an original TM image and the corresponding classified land cover map. Four major land cover types: Water, Wetlands, Grasslands, Rock/Sand, are identified in the study area according to the USGS land cover / land use references. Each type is represented by a class in the image with value 1, 2, 3, and 4, which are mapped to color blue, red, green, and brown, respectively.

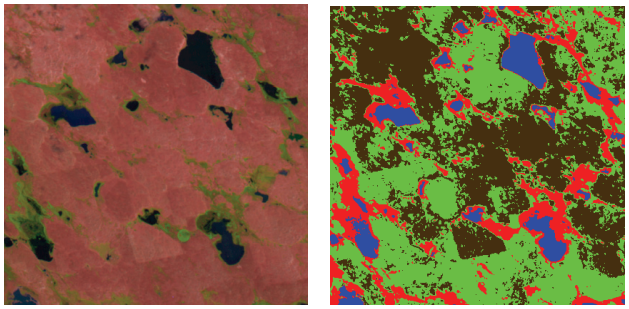

Figure 1. (a) Landsat TM image. (b) Classified land cover map.

3. REGION QUAD-TREE BASED INDEXING

The most commonly adopted method for multi-level indexing is to use a tree structure. In this study, we employ region quad-tree for image indexing. The underlying data structure in a quad-tree 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 square array of pixels into four equal-sized quadrants [14].

The region quad-tree structure takes advantage of the two-dimensional nature of the image and generates hierarchical information management, devoting more depth to the tree only in places where necessary detail is to be queried and retrieved. This approach allows the user to focus on the region of interest and meanwhile has the option to overview the whole picture whenever needed.

An example of successive decomposition and indexing of the quadrants of an image is shown in Figure 2. Assume the original image is at the default level zero, some typical indexes are illustrated in the image and the corresponding tree nodes. The depth of the tree is application specific and may be decided on the fly.

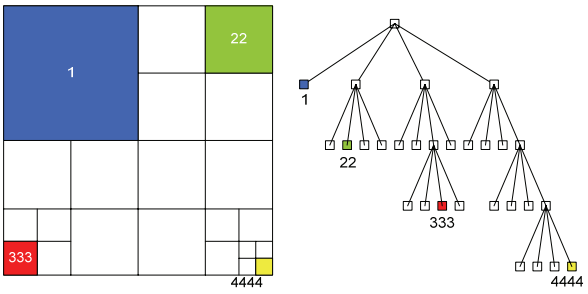

Figure 2. Multi-level image indexing by region quad-tree.

Traditional region quad-tree decomposition typically stores the image with different resolution repetitively at each level. This is effective for quick browsing but is not practical for a large collection of remote sensing images with hundreds of mega bytes each because of the significant increases of the storage overhead. Applying an appropriate compression algorithm will reduce the size but may cause loss of information.

In this study, the region quad-tree based image object created during the indexing process contains the raw data only once, while the nodes at each level maintain the references (coordinates, statistics, etc) to the original image. The image regions will be created dynamically for browsing or presentation in the query results.

4. OO DATABASES AND NATIVE QUERIES

Basically, the objects used by OO programs are analogous to entities used by OODBs, with one major difference: program objects disappear once a program stops running; database objects must exist. The idea that an object continues to exist once the program that created it has finished running is known as persistence [15]. Object-oriented database management systems (OODBMSs) are based on the concept of persistent objects and use class declarations similar to those used by OO programming languages. Relationships between objects in OODBs are built upon the class diagram just like in OO programming languages.

OODBMSs such as ObjectStore, Objectivity/DB, and Versant are not as suited to ad hoc querying as relational databases, although ODMG has been working on a standard Object Query Language (OQL) for many years [16]. No universally accepted query language exists except basic SQL-like queries that must follow predefined relationships and cannot insert new relationships on the fly. Most modern query by image content (QBIC) systems, therefore, usually do not have a general query interface but use a domain-specific and inflexible query-by-example (QbE) approach.

In this study, we adopt db4o from db4object Inc. to implement the image database. Database benchmarks show it is much faster than Hibernate and MySQL, a popular object-relational mapper and relational SQL database stack. It is currently the only OODBMS that is native to both Java and .NET, offering a wide array of unique OO database functionalities like replication, native queries (NQs), and the object manager for browsing object databases. It is also the first to implement NQs that provide database querying with OO programming language semantics [17], validated by Microsoft's LINQ (.NET Language Integrated Queries) project.

Rather than using string-based APIs such as SQL and OQL, NQs express database queries with the programming language itself (e.g., Java, C#, or VB .NET) to access the database and thus avoid a constant switch between programming language and data access API. Native Queries are fully checked at compile time and can be written using IDE auto-completion and automatically refactored by the IDE. Because native queries use the semantics of the programming language, they are standardized and a safe choice for the future.

The following shows how a simple native query to retrieve image (Raster) object by date looks like in C#. It takes the advantage of the OO programming language feature delegate in this case and uses the native logical operators.

```
public IList<Raster> ImgByDate(DateTime date,
                               int level)
{
  IList<Raster> imgList = db.Query<Raster>
    (delegate(Raster img)
    {return img.Date==date && img.Level==level;});
  return imgList;
}
```

NQs have the ability to run one or more lines of code against all instances of a class. The expressions return true to mark specific instances as part of the result set. db4o optimizes NQs expressions and runs them against indexes without instantiating actual objects. For example, to retrieve land cover distribution, a new query class is derived from the db4o Predicate class. Note how the programming interface is simplified by the definition of this query class and how complex queries (only AND logic is presented) are composed in a single Match method. A list of NQs designed and implemented in this study is shown in Figure 3.

```
public IList<Raster> ImgByClsPct(List<LandCover>
              clsList, LogicOp logicOp, int level)
{
   List<Raster> imgList = new List<Raster>();
   ObjectSet results = db.Query(new
            ClsPctQuery(clsList, logicOp, level));
  while (results.HasNext())
    imgList.Add((Raster) results.Next());
  return imgList;
}

public class ClsPctQuery : Predicate
{
  private List<LandCover> clsList;
  private LogicOp logicOp;  private int level;

  public ClsPctQuery(List<LandCover> clsList,
                LogicOp logicOp, int level)
  { this.clsList = clsList;
    this.logicOp = logicOp; this.level = level; }

  public bool Match(Raster img)
  {
    if (logicOp==LogicOp.AND && img.Level==level)
    {
      foreach (LandCover cls in clsList)
       for (int i = 0; i < img.ClsList.Count; i++)
         if (cls.clsName == img.ClsList[i].clsName
           && img.Hist[i] / img.Size < cls.pct)
         return false;
      return true;
    }...
  }
}
```

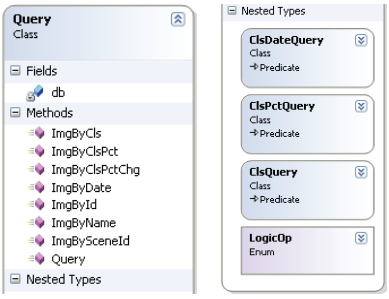

Figure 3. The query class diagram.

5. EXPERIMENTAL RESULTS

A prototype system was designed and implemented with Microsoft Visual C#, .NET 2.0 Framework, and db4o 5.2 for C#. The experiments were conducted on a time series (from 1989 to 1997) of geometric rectified Landsat TM images, covering the scenes of Sand Hills region in the Great Central Plains. Images were pre-calibrated and registered to UTM-13 map. The experimental database contains 14 images of 768 × 768 pixels each, classified and indexed to up to the depth of 2. Total number of image regions, therefore, is 14 × (1 + 4 + 16) = 294.

The system allows quick browsing of all the images in the database with a tree presentation as shown in Figure 4. The user can select the image and click on the *Display Image* button to view the image at different level. To add a new image object into the database, the user loads an image file and the system performs the classification and indexing operations.

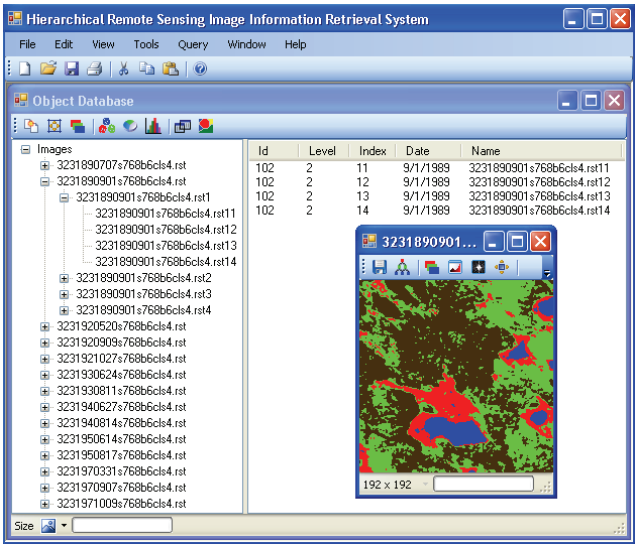

Figure 4. A tree-list view in the prototype system.

5.1. Land Cover Distribution Information Retrieval

Our first objective is to retrieve the land cover distribution information. Figure 5 (a) shows a typical query, which allows the user to select different percentage of land cover types with logical combinations like *And, Or, Not*. The user may also specify the searching depth in the quad-tree. For instance, the user is interested in finding all the regions that contain at least 10% water and 40% grasslands at level 2.

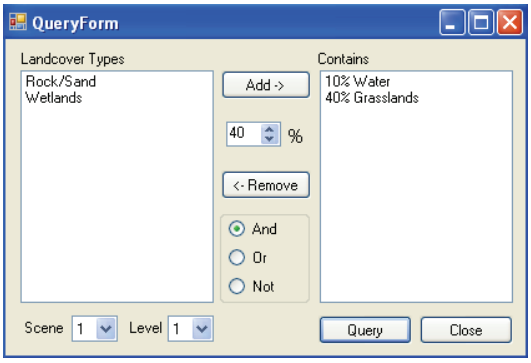

Figure 5. (a) The land cover distribution query form.

Figure 5 (b) shows the retrieved image thumbnails. The user can simply double-click on the thumbnails to view the full-size images.

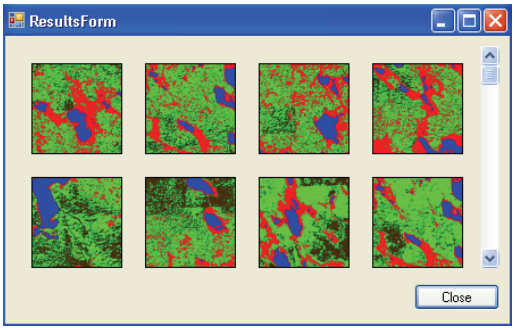

Figure 5. (b) Retrieved images displayed in thumbnails.

5.2. Land Cover Change Detection

Our second task is to find seasonal land cover changes. Figure 6 (a) is the dialog that allows the user to choose the land cover types, specify the date, and select level of details. The results are presented in a table as shown in Figure 6 (b). For example, about 82% wetlands remained and 10% turned into grasslands. The user can change the index (e.g. 3) to look into the variations in each individual region.

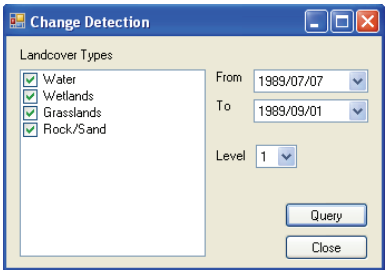

Figure 6. (a) The change detection query form.

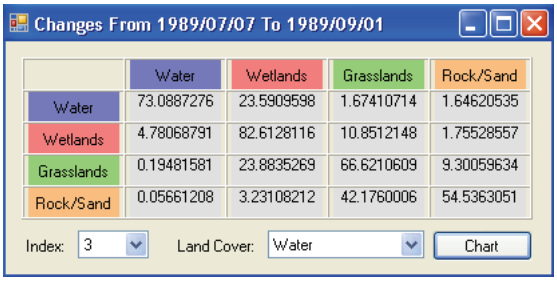

Figure 6. (b) The land cover change matrix for each region.

Figure 6 (c) shows the corresponding images. The change of grasslands and sand/rock types can be clearly observed.

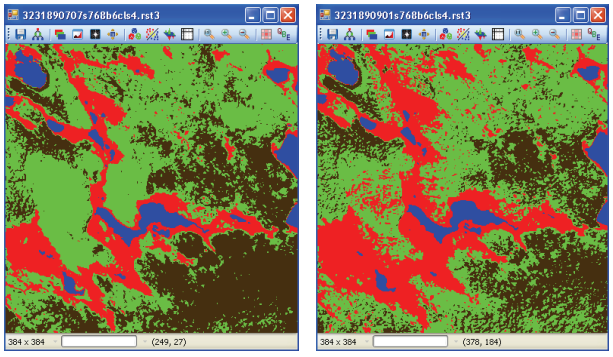

Figure 6. (c) Land cover changes from Jul. to Sep. in 1989.

Figure 7 (a) and (b) shows a query example at level 2 in fall 1997, which yields 16 regions with index 11 through 44.

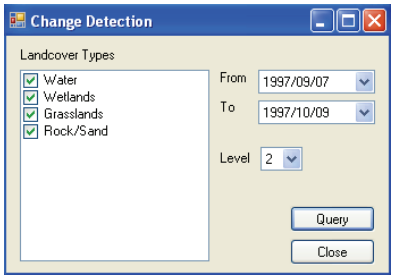

Figure 7. (a) Change detection query at level 2 in 1997.

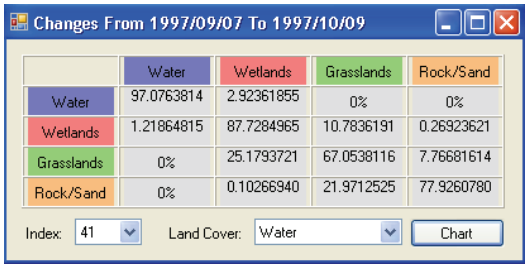

Figure 7. (b) The land cover change at level 2 in 1997.

Figure 8 (a) through (d) display the charts that compare the changes of land cover types in each region. It not only presents useful statistical information, but helps the user quickly identify the abnormal pattern in a specific region. For instance, Figure 8 (a) shows dramatic water fluctuation in region 44. It also can be observed that no water coverage was changed to grasslands or rock/sand.

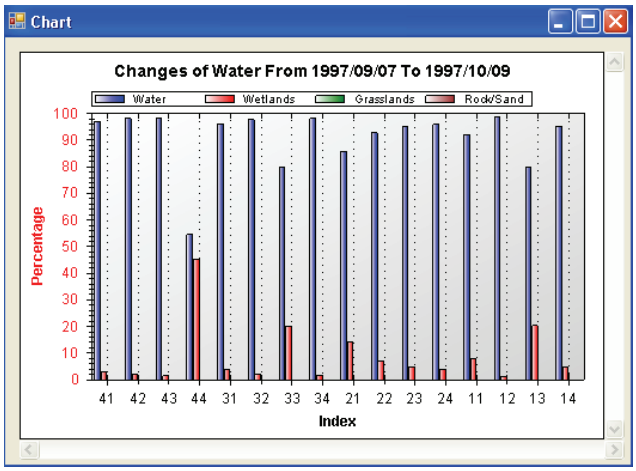

Figure 8. (a) Regional variation of water (Sep – Oct, 1997).

Figure 8 (b) shows relatively stable variations of wetlands. It is the only type that had changes to all other three types, with more to grasslands than water and rock/sand.

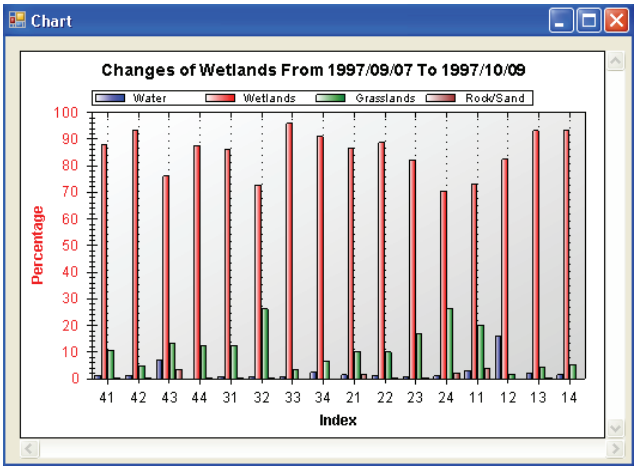

Figure 8. (b) Regional variation of wetlands (Sep – Oct, 1997).

Similarly, Figure 8 (c) and (d) show that no change from grasslands or rock/sand to water occurred. Furthermore, note in Figure 8 (d) the considerable variations from rock/sand to grasslands in regions 31, 32, 33, and 34. This indicates that the entire region with index 3 at a higher level, i.e., level 1, appears to have this pattern as well.

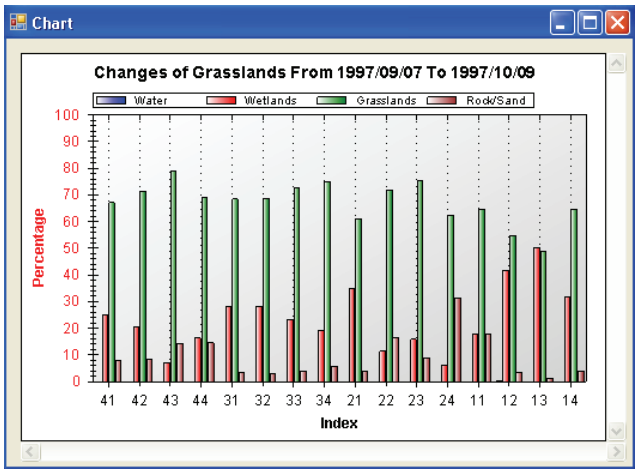

Figure 8. (c) Regional variation of grasslands (Sep – Oct, 1997).

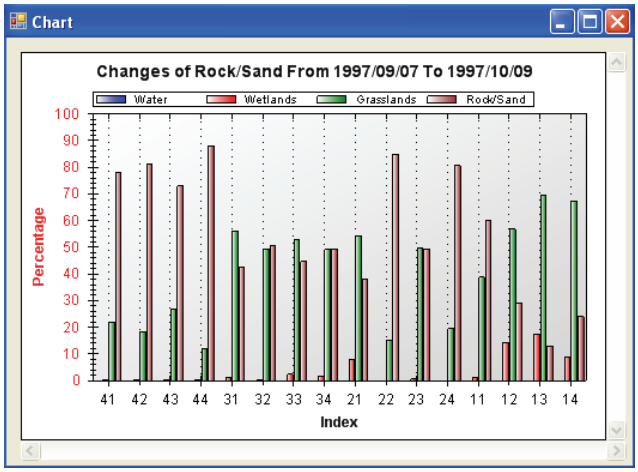

Figure 8. (d) Regional variation of rock/sand (Sep – Oct, 1997).

6. CONCLUSIONS AND FUTURE WORK

We have presented a hierarchical framework for remote sensing land cover information retrieval and change detection using a region quad-tree based indexing approach and object-oriented database model with native queries support. The experimental results show the effectiveness of the framework and the potentials in other remote sensing applications like urban planning, drought monitoring, etc.

Our future work will build a practical database that allows the user to select different scenes and compose more complex queries for trend analysis and temporal pattern retrieval in the aforementioned applications. A complexity and performance analysis of the framework will be done on the queries and database update.

We will also consider query optimization with the next db4o release, which also supports a new server side cursor technology for deterministic response times when querying in Client/Server multi-user environments.

7. REFERENCES

[1] Gorte, B. and Stein, A. Bayesian classification and class area estimation of satellite images using stratification. *IEEE Trans Geosci. Rem. Sens., 36, 3* (May 1998), 803 – 812.

[2] Mantero, P., Moser, G., and Serpico, S.B. Partially supervised classification of remote sensing images through SVM-based probability density estimation. *IEEE Trans Geosci. Rem. Sens., 43, 3* (Mar. 2005), 559 – 570.

[3] Weisberg, A., Najarian, M., Borowski, B., Lisowski, J., and Miller, B. Spectral angle automatic cluster routine (SAALT): an unsupervised multispectral clustering algorithm. In *Proc. of 1999 IEEE Aerospace Conf.*, (Aspen, CO, USA, March 6 – 13, 1999). vol. 4, 307 – 317.

[4] Pesaresi, M. and Benediktsson, J.A. A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE Trans. on Geosci. Rem. Sens., 39, 2* (Feb. 2001), 309 – 320.

[5] Selim, S.Z. and Ismail, M.A. *K*-means-type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Trans. Pattern Anal. Mach. Intell., 6, 1* (Jan. 1984), 81 – 87.

[6] Kanungo, T., Mount, D.M., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A.Y. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell., 24, 7* (Jul. 2002), 881 – 892.

[7] Huang, J.Z., Ng, M.K., Rong, H., and Li, Z. Automated variable weighting in *k*-means type clustering. *IEEE Trans. Pattern Anal. Mach. Intell., 27, 5* (May 2005), 657 – 668.

[8] Samet, H. Region representation: quadtrees from binary arrays. *Computer Graphics and Image Processing, 13, 1* (May 1980), 88 – 93.

[9] Cook, W.R. and Rosenberger, C. *Native Queries for Persistent Objects, A Design White Paper*. db4object Inc., August, 2005.

[10] Ray, S. and Turi, R. H. Determination of number of clusters in *k*-means clustering and application in color image segmentation. In *Proc. of the 4th Inter. Conf. on Adv. in Patt. Recog. Dig. Tech.*, (Calcutta, India, Dec. 1999), 137 – 143.

[11] Bradley, P.S. and Fayyad, U.M. Refining initial points for *k*-means clustering. In *Proc. of the 15th Inter. Conf. on Machine Learning*, (San Francisco, CA, 1998), 91 – 99.

[12] Pena, J.M. and Larranaga, P. An empirical comparison of four initialization methods for the *k*-means algorithm. *Pattern Recognition Letters, 20* (1999), 1027 – 1040.

[13] Li, J. and Narayanan, R.M. Integrated spectral and spatial information mining in remote sensing. *IEEE Trans. Geosci. Rem. Sens., 42, 3* (Mar. 2004), 673 – 685.

[14] Samet, H., and Webber, R. Storing a collection of polygons using quadtrees. *ACM Trans. on Graphics, 4, 3* (Jul. 1985), 182 – 222.

[15] Atkinson, M. P. and Morrison, R. Orthogonally persistent object systems. *Int. J. Very Large Data Bases 4, 3* (1995), 319 – 401.

[16] Subieta, K. Object-oriented standards: Can ODMG OQL be extended to a programming language?. In *Proc. of the International Symposium on Cooperative Database Systems for Advanced Applications*, (Japan, 1996), 546 – 555.

[17] Cook, W.R. and Rai, S. Safe query objects: statically typed objects as remotely executable queries. In *Proc. 27th Inter. Conf. on Software Engineering, ICSE 2005*, (St. Louis, MO, USA, May 15 – 21, 2005), 97 – 106.